UNIVERSIDAD PÚBLICA DE NAVARRA

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa

DEPARTAMENTO DE AUTOMÁTICA Y COMPUTACIÓN

Novel methodologies for improving fuzzy classifiers: dealing with multi-class and Big Data classification problems

Mikel Elkano Ilintxeta

TESIS DOCTORAL

Pamplona, marzo de 2018

UNIVERSIDAD PÚBLICA DE NAVARRA

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa

DEPARTAMENTO DE AUTOMÁTICA Y COMPUTACIÓN

Novel methodologies for improving fuzzy classifiers: dealing with multi-class and Big Data classification problems

Memoria presentada por

Mikel Elkano Ilintxeta

para optar al grado de Doctor por

la Universidad Pública de Navarra

Pamplona, marzo de 2018

Autorización

Dr. Mikel Galar Idoate: Profesor ayudante doctor de la Universidad Pública de Navarra en el Área de Conocimiento de Ciencias de la Computación e Inteligencia Artificial

Dra. Edurne Barrenechea Tartas: Profesora titular de la Universidad Pública de Navarra en el Área de Conocimiento de Ciencias de la Computación e Inteligencia Artificial

HACEN CONSTAR que el presente trabajo titulado "*Novel methodologies for improving fuzzy classifiers: dealing with multi-class and Big Data classification problems*" ha sido realizado bajo su dirección por D. Mikel Elkano Ilintxeta.

Autorizándole a presentarlo como Memoria para optar al grado de Doctor por la Universidad Pública de Navarra.

El Doctorando

Fdo: Mikel Elkano Ilintxeta

Los Directores

Fdo: Mikel Galar Idoate

Fdo: Edurne Barrenechea Tartas

Pamplona, marzo de 2018

Agradecimientos

Tres años y medio y unos cuantos “tenemos que darle otra vuelta a ésto” después, ha llegado la hora de concluir esta tesis doctoral. Cuando miro atrás y visualizo el largo camino recorrido, no puedo evitar preguntarme: “¿volverías a hacerlo?”. Salvo en momentos de frustración, no suelo invertir demasiado tiempo en responder a esa pregunta. Siempre he querido ser investigador. Por ello, quiero comenzar estas líneas de agradecimiento expresando mi gratitud al grupo de investigación que me ha permitido vivir de esta apasionante profesión durante estos años: el Grupo de Investigación de Inteligencia Artificial y Razonamiento Aproximado (GIARA) de la Universidad Pública de Navarra. Especialmente me gustaría dar las gracias al creador y director de GIARA, Humberto Bustince, por todo el apoyo y la confianza depositada en mí durante la realización de la tesis. Por supuesto, mis directores ocupan un lugar especial en estos agradecimientos. Ellos son los que han sufrido mi carácter en momentos difíciles y los que me han ayudado a reconducir el trabajo cuando ha sido necesario. No solo tienen un talento indiscutible, sino que además transmiten una ética de trabajo envidiable. A pesar de que no consta como director de tesis, quiero agradecer expresamente a José Antonio Sanz Delgado el tiempo que ha dedicado a todos y cada uno de nuestros trabajos. Él y Mikel Galar Idoate fueron los tutores de mi proyecto fin de carrera y han sido mis mentores y amigos desde hace más de cuatro años.

Y qué puedo decir sobre mis dos familias (Ilintxeta y Elkano). Siempre han mostrado interés y han estado pendientes de mi progreso, tanto en la carrera como en el doctorado. Ese apoyo incondicional es fundamental para afrontar nuevos retos. Me considero un afortunado por tener una familia tan unida. Como anécdota, me gustaría aclarar que la firma que empleo en las publicaciones solo incluye el apellido Elkano porque desconocía la posibilidad de añadir un guión para que consten ambos apellidos. Sin duda, la firma más adecuada sería Elkano-Ilintxeta.

Aunque no sean Elkano-Ilintxeta, siempre he dicho que Sergio Bayascas y Mikel Adot son como mis hermanos. Ellos han sido testigos de todos los pasos que he dado desde adolescente y han sido fieles acompañantes en todas mis aventuras. Creo que saben de sobra que cualquier párrafo que pudiera escribir sobre ellos se quedaría corto. Y por supuesto incluyo en este párrafo a Erika Escobar, una de las mejores personas que conozco. Erika cambió mi vida hace cinco años y medio y siempre estaré agradecido por todo lo que ha hecho por mi.

Y ha llegado el turno de mis *aitas*. Qué puedo decir: objetivo cumplido. Ellos saben que este doctorado también es suyo. Desde *txiki* me han oído decir que yo quería ser uno de esos investigadores con bata blanca que están en esos laboratorios tan emocionantes. A pesar de que no haya batas ni probetas por medio, mi objetivo final no ha cambiado: intentar ayudar a lo

sociedad investigando nuevas soluciones que contribuyan al progreso. Mis *aitas* saben que la bata y la probeta no eran más que objetos que simbolizan ese objetivo. Y ese espíritu universitario y altruista que me han transmitido siempre es el que me ha traído hasta este punto.

Eskerrik asko guztioi. Muchas gracias por todo.

Índice

I. Memoria **3**

1. Introducción . . . 3
2. Planteamiento del problema . . . 7
   - 2.1. Clasificación . . . 7
   - 2.2. Sistemas de Clasificación Basados en Reglas Difusas . . . 8
     - 2.2.1. Algoritmo de Chi et al. . . . 11
     - 2.2.2. FARC-HD . . . 11
   - 2.3. Problemas multi-clase . . . 12
     - 2.3.1. Estrategia uno-contra-uno . . . 13
     - 2.3.2. Estrategia uno-contra-todos . . . 16
   - 2.4. Problemas de clasificación en Big Data . . . 16
     - 2.4.1. Apache Hadoop . . . 18
     - 2.4.2. Apache Spark . . . 19
     - 2.4.3. Clasificadores difusos en Big Data . . . 20
     - 2.4.4. Reducción de prototipos en Big Data . . . 22
3. Motivación . . . 23
4. Objetivos . . . 25
5. Discusión de resultados . . . 26
   - 5.1. Mejorando la clasificación multi-clase en el clasificador difuso FARC-HD: sobre la sinergia de funciones de solapamiento n-dimensionales y estrategias de descomposición . . . 26
   - 5.2. Sistemas de clasificación basados en reglas difusas para problemas multi-clase utilizando estrategias de descomposición binaria: sobre la influencia de las funciones de solapamiento n-dimensionales en el método de razonamiento difuso . . . 28

5.3. CHI-BD: un nuevo sistema de clasificación basado en reglas difusas para problemas de clasificación Big Data . . . 30
5.4. CHI-PG: algoritmo para la generación rápida de prototipos en problemas de clasificación Big Data . . . 32
5.5. CFM-BD: algoritmo distribuido de inducción de reglas para la construcción de Modelos Difusos Compactos en problemas de clasificación Big Data . . . 33
6. Conclusiones . . . 35
7. Líneas futuras . . . 36
8. Introduction and conclusions (English version) . . . 39
8.1. Introduction . . . 39
8.2. Conclusions . . . 42

Bibliografía **45**

II. Publicaciones: trabajos publicados, aceptados y sometidos **55**

1. Mejorando la clasificación multi-clase en el clasificador difuso FARC-HD: sobre la sinergia de funciones de solapamiento n-dimensionales y estrategias de descomposición – *Enhancing Multiclass Classification in FARC-HD Fuzzy Classifier: On the Synergy Between n-Dimensional Overlap Functions and Decomposition Strategies* . 55
2. Sistemas de clasificación basados en reglas difusas para problemas multi-clase utilizando estrategias de descomposición binaria: sobre la influencia de las funciones de solapamiento n-dimensionales en el método de razonamiento difuso – *Fuzzy Rule-Based Classification Systems for multi-class problems using binary decomposition strategies: On the influence of n-dimensional overlap functions in the Fuzzy Reasoning Method* . . . 77
3. CHI-BD: un nuevo sistema de clasificación basado en reglas difusas para problemas de clasificación Big Data – *CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems* . . . 101
4. CHI-PG: algoritmo para la generación rápida de prototipos en problemas de clasificación Big Data – *CHI-PG: A fast prototype generation algorithm for Big Data classification problems* . . . 131
5. CFM-BD: algoritmo distribuido de inducción de reglas para la construcción de Modelos Difusos Compactos en problemas de clasificación Big Data – *CFM-BD: a distributed rule induction algorithm for building Compact Fuzzy Models in Big Data classification problems* . . . 167

Parte I. Memoria

1. Introducción

Las conductas más primitivas del ser humano están determinadas en gran medida por modelos de *expectativa-recompensa* guiados en su mayoría por circuitos y sistemas dopaminérgicos [Gli11]. El cerebro humano está continuamente realizando predicciones sobre su entorno en base a la información que recibe de los diferentes sistemas sensoriales. Cuando los estímulos llegan al cerebro, éste los procesa con el objetivo de dar una respuesta que maximice la recompensa. Por ejemplo, si pensamos en el acto de agarrar una manzana para llevárnosla a la boca, la información recibida (principalmente) de los ojos y de los mecanoreceptores de la mano se procesa para aplicar una presión adecuada en una cierta localización. En este caso, la respuesta sería la acción de coger la manzana e introducirla en la boca y la recompensa vendría dada por los nutrientes adquiridos gracias a dicha acción. Sin embargo, el cerebro no puede evaluar la recompensa obtenida antes de haber llevado a cabo la acción propiamente dicha. Por tanto, necesita un mecanismo de *predicción* que anticipe de manera aproximada la recompensa asociada a una determinada acción. A esta predicción la llamamos *expectativa*. De esta forma, el éxito de un sujeto en este contexto vendrá dado por la similitud entre la recompensa obtenida y la esperada. A lo largo del tiempo, el cerebro ajusta este modelo predictivo en base a los errores cometidos en el pasado (experiencia) y modula las respuestas en consecuencia. A este proceso de adaptación lo llamamos *aprendizaje*.

Una de las ramas de conocimiento existentes en el campo de la Inteligencia Artificial es el *aprendizaje automático*, en inglés *machine learning* [Alp04]. Las contribuciones en esta área de conocimiento intentan crear modelos matemáticos y computacionales que permitan a las máquinas construir sistemas predictivos de manera automática. De forma similar al modelo de *expectativa-recompensa*, la máquina realiza predicciones en base a la información entrante y evalúa el error cometido, el cual se usará para aprender de la experiencia y adaptar el modelo predictivo. En aprendizaje automático, existen diferentes tipos de predicciones según su naturaleza, como por ejemplo la *clasificación* y la *regresión*. En las tareas de clasificación, el sistema (denominado *clasificador*) identifica y clasifica el estímulo (datos) de entrada y predice la clase a la que pertenece dicho estímulo. Por ejemplo, un coche de conducción autónoma debe ser capaz de identificar las señales de Stop en las imágenes que recibe por las cámaras [ZLZ+16]. Una herramienta automática de diagnóstico médico es capaz de identificar y clasificar el perfil cognitivo de un paciente de Parkinson en base a su historial y a las señales de electroencefalograma [MSFI+14]. En ambos ejemplos, la predicción del sistema es una categoría, clase, o concepto. En las tareas de regresión, la predicción realizada por el sistema es de naturaleza continua. Siguiendo con el ejemplo del coche autónomo, una vez el coche ha detectado la señal de Stop, éste debe aplicar una cierta fuerza a los frenos del coche para conseguir un ritmo de frenado adecuado y así, detener el vehículo con tiempo suficiente evitando "frenazos" bruscos. La predicción en este caso es el módulo de la fuerza de frenado, es decir, un número real.

Además de los diferentes tipos de predicciones, existen varias formas de aprendizaje automático, entre las que se encuentran el aprendizaje *supervisado*, *no-supervisado*, y por *refuerzo*. Tanto en el aprendizaje supervisado como en el no-supervisado, el sistema dispone de un *conjunto de datos*

de entrenamiento (experiencia) formado por una serie de *ejemplos* (también llamados instancias o transacciones). En el ejemplo del Parkinson, un ejemplo representaría los datos recogidos sobre un paciente determinado. La diferencia entre estos dos tipos de aprendizaje es que en la vertiente supervisada el sistema conoce la predicción (salida) adecuada para cada ejemplo de entrenamiento, mientras que en las tareas no-supervisadas el sistema desconoce dicha salida. Esta salida ideal es proporcionada por un humano denominado *experto*. De esta forma, los métodos de aprendizaje supervisados consisten en ajustar un modelo predictivo que minimice el error cometido respecto al conjunto de entrenamiento, al mismo tiempo que maximice la capacidad de generalización. Esta capacidad de generalización es fundamental para que el modelo realice predicciones certeras cuando recibe ejemplos desconocidos. En el caso de las tareas no-supervisadas, el objetivo del aprendizaje es descubrir y extraer las características y propiedades que determinan la estructura interna de los ejemplos de entrenamiento. Al igual que en los métodos supervisados, la capacidad de generalización del sistema es vital para que la estructura aprendida siga siendo válida en ejemplos futuros. Finalmente, el aprendizaje por refuerzo sustituye (o complementa) el conjunto de entrenamiento por estímulos recibidos del entorno, que son utilizados como una señal de retorno o *feedback* que modula y ajusta las respuestas futuras del sistema.

En esta tesis planteamos metodologías que están enfocadas únicamente a tareas de clasificación supervisadas. Existe una gran variedad de algoritmos capaces de resolver estas tareas, cada uno con métodos de aprendizaje y modelos predictivos diferentes. Nosotros nos centramos en uno de los más populares: los Sistemas de Clasificación Basados en Reglas Difusas (SCBRDs) [INN04]. La ventaja de estos sistemas es que proporcionan un modelo formado por una serie de reglas que contienen etiquetas lingüísticas interpretables por el ser humano, lo que les permite explicar el razonamiento llevado a cabo al realizar una predicción. Por ejemplo, si se entrena un SCBRDs para discernir entre diferentes perfiles de deterioro cognitivo en pacientes con Alzheimer, podrían obtenerse una serie de reglas del tipo "si el paciente es mayor y los niveles de proteína *tau* son muy altos, entonces el paciente sufre demencia" ó "si el paciente es joven y los niveles de proteína tau son altos, entonces el paciente sufre deterioro cognitivo leve". Nótese que las reglas contienen términos como "mayor", "joven", "muy alto", ó "alto", conceptos que son imprecisos per se. Estas etiquetas lingüísticas permiten a los SCBRDs no solamente explicar el porqué de las predicciones, sino también manejar la incertidumbre proveniente de información imprecisa. Gracias a estas propiedades, este tipo de técnicas se han empleado en multitud de aplicaciones del mundo real, incluyendo la bioinformática [HHCH06], medicina [SGJ$^+$13], seguridad cibernética [TKW07], finanzas [SBH$^+$14], procesamiento de imagen [NSY07], y la predicción de congestión de tráfico [ZOP$^+$14].

Los problemas de clasificación pueden dividirse en dos grupos dependiendo del número de clases que los componen: binarios (dos clases) y multi-clase (más de dos clases). En general, los problemas multi-clase implican fronteras de decisión más complejas que son más difíciles de aprender que en problemas binarios, debido al mayor número de clases. Una forma eficaz de lidiar con esta situación es descomponer el problema multi-clase original en problemas binarios más sencillos [GFB$^+$11,LCG08]. Posteriormente se entrena un clasificador independiente por cada uno de los problemas, de tal forma que cada clasificador se especializa en discernir entre dos clases únicamente. A la hora de clasificar un nuevo ejemplo, cada uno de estos clasificadores realiza una

predicción en base a su experiencia y todas estas predicciones se agregan para tomar una decisión consensuada. Esta metodología ha permitido mejorar el rendimiento de clasificadores que tratan el problema multi-clase directamente [FÖ2, RK04, GFB+11, GFBH14] y ha mostrado ser eficaz a la hora de mejorar el rendimiento de los SCBRDs previamente mencionados [IYN05, HB08, HH09a, FCB+10, SH11]. Sin embargo, el uso de estrategias de descomposición en SCBRDs plantea una nueva problemática: lidiar con diferentes estructuras de reglas y *métodos de razonamiento difuso* (FRM). Las diferencias estructurales en las reglas vienen dadas por la variedad de métodos de construcción de reglas existentes en la literatura. Estos métodos pueden diferir, por ejemplo, en el tipo de etiquetas lingüísticas generadas, en el operador de conjunción/disyunción empleado en reglas con más de un antecedente, o en la longitud media de las reglas. Por otro lado, el FRM encargado de inferir la salida adecuada a partir de las reglas construidas puede variar notablemente de un SCBRD a otro. Estos factores hacen que el comportamiento de las técnicas de descomposición sea dependiente del SCBRD empleado. Por consiguiente, algunos de los métodos de agregación más populares no son capaces de aprovechar el potencial mostrado en otro tipo de clasificadores.

Además de la dificultad añadida de los problemas multi-clase, en los últimos años las técnicas de aprendizaje automático se han topado con un nuevo reto: la ingente y creciente cantidad de información producida y consumida por el ser humano, también conocida como *Big Data*. De acuerdo con el famoso estudio realizado por Gantz y Reinsel en el año 2012 [GR12], esta cantidad se duplicaría cada dos años entre 2012 y 2020. A pesar de haberse planteado una gran variedad de definiciones del término Big Data [Art13, GH15, NMS+15], en esta tesis consideramos que los problemas Big Data surgen cuando la cantidad de información a procesar excede la capacidad de cómputo o almacenamiento de un ordenador convencional moderno. No obstante, no existe ningún umbral universal a partir del cual una cierta cantidad de información es considerada como Big Data, ya que dependerá del tipo de tarea que se lleve a cabo con dicha información. Por ejemplo, la exigencia computacional de métodos estadísticos descriptivos clásicos como el cálculo de medias, desviaciones, histogramas, etc. es mucho menor que la de los métodos de aprendizaje automático. Si consideramos un historial de pacientes de 10GB, la tarea de obtener la edad media de esos pacientes podría ser viable en un período de tiempo razonable, mientras que la extracción de patrones farmacológicos podría requerir un período de tiempo inaceptable. Además, un ordenador convencional equipado con 8GB de memoria RAM no sería capaz de almacenar toda la información disponible en la memoria principal y ni siquiera podría ejecutar el algoritmo. En este nuevo escenario, los métodos de aprendizaje automático que venían utilizándose hasta la fecha ya no son viables. Una de las soluciones más populares para trabajar en entornos Big Data es la *computación distribuida* [GGL03, DG08]. Esta solución consiste en dividir el conjunto de datos original en múltiples subconjuntos que se distribuyen a través de varios *nodos*. Un nodo es simplemente un ordenador, pudiendo ser tanto un PC de sobremesa convencional como un servidor especializado. Estos nodos se conectan entre sí formando una red (generalmente local) a la que llamamos *cluster*. Cuando se procesa un conjunto de datos, cada uno de estos nodos procesa únicamente el subconjunto de datos que le ha sido asignado. Posteriormente, todos los resultados parciales generados en los nodos se agregan y se obtiene el resultado final. A pesar de que esta metodología soluciona los problemas asociados con las exigencias de cómputo y almacenamiento, el procesamiento distribuido de la información implica diseñar métodos que

soporten dicha funcionalidad. En el caso de los SCBRDs diseñados para Big Data, la dificultad añadida de la computación distribuida ha impedido explotar el potencial que han mostrado estos sistemas cuando se han aplicado de forma local y secuencial.

Otra metodología (complementaria) para poder manejar grandes volúmenes de datos son las técnicas de reducción de prototipos (PR) [GDCH12,NL11]. El objetivo de este tipo de técnicas es construir una versión reducida del conjunto de datos de entrenamiento que mejore las predicciones en futuros ejemplos y minimice el número de ejemplos que contiene el conjunto reducido [KG15]. En otras palabras, los métodos de PR permiten que algoritmos de aprendizaje automático que no están diseñados para Big Data puedan ejecutarse en estos entornos empleando una versión reducida de los datos. Además, diversos estudios han mostrado que este proceso de reducción ayuda a mejorar la precisión de estos algoritmos incluso en problemas que no son Big Data [GLH14]. Sin embargo, gran parte de las aproximaciones de PR propuestas hasta la fecha presentan serias limitaciones de escalabilidad que afectan a su eficiencia. Paradójicamente, el exceso de requerimientos de cómputo y de almacenamiento que caracterizan a los algoritmos de aprendizaje automático están también presentes en muchas de las técnicas de PR existentes [GLH14]. Por consiguiente, el beneficio obtenido en términos de tiempo y memoria a la hora de aprender/clasificar puede ser eclipsado por los propios requerimientos del proceso de reducción, si bien la precisión del clasificador puede seguir beneficiándose de dicho proceso.

El objetivo de esta tesis es presentar nuevas metodologías para mejorar el rendimiento de los SCBRDs en los dos escenarios presentados anteriormente: multi-clasificación y Big Data. Para ello, nos hemos centrado en el diseño de tres tipos de soluciones, uno enfocado a problemas multi-clase y los otros dos a entornos Big Data. En el caso de los problemas multi-clase, hemos estudiado y analizado el efecto de diferentes métodos de aprendizaje y razonamiento difuso de varios SCBRDs en el rendimiento de las estrategias de descomposición. Una vez identificados algunos de los problemas que presenta esta sinergia, hemos propuesto una modificación del FRM que permite mejorar su rendimiento. En cuanto a las metodologías planteadas para Big Data, hemos presentado dos nuevos algoritmos de aprendizaje distribuido para SCBRDs que solucionan algunas de las limitaciones presentes en los métodos existentes. De forma transversal, hemos aprovechado uno de estos algoritmos para desarrollar un nuevo método de PR de complejidad lineal. Para describir las soluciones propuestas, la memoria se ha dividido en dos partes:

- Parte I. Dedicada al planteamiento del problema, a la discusión de los métodos propuestos y a las conclusiones obtenidas.
- Parte II. Contiene las publicaciones asociadas a la tesis.

En la Parte I, comenzamos planteando formalmente el problema y presentando las técnicas utilizadas (Sección 2), las problemática concreta que nos ha llevado a la elaboración de esta tesis (Sección 3), y los objetivos de la misma (Sección 4). Posteriormente resumimos los trabajos realizados y discutimos los resultados más relevantes (Sección 5). Finalmente, presentamos las conclusiones de la memoria (Sección 6) e introducimos algunas de las líneas futuras que surgen a partir de las investigaciones realizadas (Sección 7).

La Parte II presenta un compendio de cinco publicaciones asociadas a esta tesis:

- Mejorando la clasificación multi-clase en el clasificador difuso FARC-HD: sobre la sinergia de funciones de solapamiento n-dimensionales y estrategias de descomposición – *Enhancing Multiclass Classification in FARC-HD Fuzzy Classifier: On the Synergy Between n-Dimensional Overlap Functions and Decomposition Strategies*

- Sistemas de clasificación basados en reglas difusas para problemas multi-clase utilizando estrategias de descomposición binaria: sobre la influencia de las funciones de solapamiento n-dimensionales en el método de razonamiento difuso – *Fuzzy Rule-Based Classification Systems for multi-class problems using binary decomposition strategies: On the influence of n-dimensional overlap functions in the Fuzzy Reasoning Method*

- CHI-BD: un nuevo sistema de clasificación basado en reglas difusas para problemas de clasificación Big Data – *CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems*

- CHI-PG: algoritmo para la generación rápida de prototipos en problemas de clasificación Big Data – *CHI-PG: A fast prototype generation algorithm for Big Data classification problems*

- CFM-BD: algoritmo distribuido de inducción de reglas para la construcción de Modelos Difusos Compactos en problemas de clasificación Big Data – *CFM-BD: a distributed rule induction algorithm for building Compact Fuzzy Models in Big Data classification problems*

2. Planteamiento del problema

En esta sección describimos formalmente la problemática abordada por las metodologías presentadas en esta memoria. Las Secciones 2.1 y 2.2 repasan los conceptos de las tareas de clasificación supervisada y de los Sistemas de Clasificación Basados en Reglas Difusas (SCBRDs), respectivamente. En la Sección 2.3 profundizamos en los problemas de clasificación multi-clase y las estrategias de descomposición. Finalmente, la Sección 2.4 se centra en las tareas de clasificación de grandes volúmenes de datos (Big Data).

2.1. Clasificación

Desde el punto de vista del aprendizaje supervisado, las tareas de clasificación son aquellas que, a partir de un conjunto de ejemplos de un problema concreto previamente clasificados (*conjunto de entrenamiento*), intentan construir un modelo predictivo (*clasificador*) capaz de clasificar ejemplos desconocidos. Cada uno de los ejemplos x contenidos en el conjunto de entrenamiento TR está determinado por un conjunto de F observaciones $x = (x_1, \ldots, x_F)$ denominadas variables, atributos, o características, y pertenece a una clase $y \in \mathbb{C} = \{C_1, C_2, ..., C_M\}$, donde M es el número de clases del problema. Por tanto, la construcción de un clasificador consiste en encontrar una función de decisión $h : x \rightarrow y$ óptima que maximice la bondad del clasificador, la cual es evaluada empleando técnicas de estimación del error [WK91]. Estas técnicas están basadas en el porcentaje de acierto del clasificador sobre un *conjunto de prueba* compuesto por ejemplos que

no han sido utilizados durante el proceso de aprendizaje. De esta forma, el proceso completo de construcción de un clasificador está compuesto por dos fases:

- Fase de aprendizaje o de entrenamiento: la función de decisión $h : x \rightarrow y$ se ajusta en base al conjunto de entrenamiento, obteniendo el modelo predictivo.
- Fase de prueba: el modelo obtenido se evalúa empleando ejemplos que no han sido utilizados durante el aprendizaje. En esta fase se pone a prueba la capacidad de generalización del modelo, es decir, la capacidad de clasificar correctamente ejemplos desconocidos.

Los sistemas de clasificación automática han sido empleados en una gran variedad de aplicaciones, incluyendo la bioinformática [TCM04, LX09, SMC12], el reconocimiento de patrones biométricos [HMCC08], la medicina [GU07, AS09], o el reconocimiento de imágenes [KSH12]. Estos sistemas suelen presentarse como una herramienta de soporte que ayuda, pero no reemplaza, al ser humano en la toma de decisiones. Sin embargo, los algoritmos de clasificación automática pueden, en ciertos casos, mejorar el rendimiento del ser humano en tareas de clasificación [HZRS15]. Entre las principales ventajas se encuentra el hecho de que los sistemas automáticos son capaces de procesar más información (digital) en menos tiempo, lo que les permite obtener una visión de conjunto fuera del alcance de un único ser humano. Además, un algoritmo automático no sufre distracciones, cansancio, o alteraciones emocionales que puedan afectar al rendimiento, al mismo tiempo que evita sesgos y prejuicios (subjetividad). No obstante, debemos remarcar que nos referimos únicamente al rendimiento en la clasificación de información digital, sin considerar en ningún caso cualquier otro tipo de procesamiento (capacidad) cerebral.

En la literatura existen diversos tipos de algoritmos de clasificación, como por ejemplo las *Máquinas de Soporte Vectorial* (SVM) [Vap98], los *árboles de decisión* [Qui86], los *razonamientos basado en casos* [AP94], las *redes neuronales artificiales* [Wer90], o los *Sistemas de Clasificación Basados en Reglas Difusas* (SCBRDs) [INN04]. Generalmente estos métodos difieren en varios aspectos como la precisión, la interpretabilidad del modelo, o los requerimientos computacionales y de almacenamiento. Nosotros nos centramos en los SCBRDs, los cuales se caracterizan por ofrecer un buen equilibrio entre precisión e interpretabilidad, como veremos en la siguiente sección.

2.2. Sistemas de Clasificación Basados en Reglas Difusas

Los SCBRDs son una extensión de los sistemas basados en reglas clásicos compuestos por reglas del tipo SI-ENTONCES (IF-THEN). Lo que diferencia a los SCBRDs respecto a estos últimos es que los antecedentes de las reglas están compuestos por proposiciones de la *lógica difusa* [Zad65]. La teoría de la lógica difusa proporciona un marco matemático que permite modelar la imprecisión asociada al lenguaje humano, proporcionando herramientas formales para su tratamiento. Por ejemplo, si queremos que un sistema automático aprenda a cocinar de un experto cocinero, los consejos recibidos podrían ser del tipo:

1. Cortar dos rebanadas de pan *medianas*
2. Untar *bastante* mantequilla

3. Echar *un poco* de azúcar
4. Tostar *ligeramente* el pan

Como se puede apreciar, términos como los mostrados en cursiva conllevan una cierta imprecisión. Por ejemplo, cuando decimos que una persona es alta, baja, o mediana, ¿qué rango de alturas corresponde a cada término? Si una persona midiera 1'78 metros y otra 1'80, ¿la diferencia sería suficientemente grande como para describirlos con diferentes términos? Las fronteras que delimitan todos estos términos lingüísticos no son del todo claras. Se tratan pues de fronteras *difusas*. Para representar esta incertidumbre, la lógica difusa emplea el concepto de *conjunto difuso*. Un conjunto difuso es un conjunto que hace referencia a una cierta etiqueta lingüística (e.g., alto, medio, o bajo) y que puede contener elementos de forma parcial, es decir, que un elemento puede pertenecer al conjunto con un cierto *grado de pertenencia*. Este grado de pertenencia viene dado por la *función de pertenencia* definida como $\mu_A(x) : X \rightarrow [0, 1]$, donde X representa el conjunto universal que contiene todos los valores posibles para A. De esta forma, una persona puede tener un grado de pertenencia de 0.7 al conjunto (etiqueta) *Alto* y de 0.3 al conjunto *Mediano*, y por tanto esta persona no es ni *del todo alta* ni *del todo mediana*.

Los SCBRDs tratan de construir modelos que representen el conocimiento extraído del problema empleando términos lingüísticos como los que acabamos de mencionar. Esta propiedad les permite realizar inferencias mediante métodos de razonamiento aproximados capaces de manejar información imprecisa. Gracias a estas cualidades, los SCBRDs han sido ampliamente utilizados en campos tan diversos como la bioinformática [HHCH06], la medicina [SGJ+13], la predicción de congestión de tráfico [ZOP+14], las finanzas [SBH+14], la seguridad cibernética [TKW07], o el procesamiento de imagen [NSY07].

El tipo más común de SCBRDs son los SCBRDs *lingüísticos* o de tipo *Mamdani* [Mam74], cuya estructura genérica se muestra en la Figura 1. La *Base de Conocimiento* (KB, del inglés Knowledge Base) almacena la información disponible sobre el problema en dos niveles diferentes:

- La *Base de Datos* (DB, del inglés Data Base). Contiene la definición de las funciones de pertenencia asociadas a las etiquetas lingüísticas.
- La *Base de Reglas* (RB, del inglés Rule Base). Está formada por un conjunto de reglas lingüísticas unidas por una conectiva de reglas (operador "también"), permitiendo que múltiples reglas puedan dispararse simultáneamente con la misma entrada. De entre todos los tipos de reglas definidas en la literatura especializada, en esta memoria consideramos el uso de reglas con la siguiente estructura:

  $$R_j : \text{SI } X_1 \text{ es } A_{j1} \text{ Y } \cdots \text{ Y } X_F \text{ es } A_{jF} \text{ ENTONCES Clase} = C_j \text{ con } RW_j$$

  donde R_j es la etiqueta de la regla j-ésima, $x = (x_1, \ldots, x_F)$ es un vector F-dimensional que representa el ejemplo, A_{jf} es una etiqueta lingüística, C_j es la clase asociada a la regla (consecuente), y RW_j es el peso de la regla, normalmente asociado al factor de certeza [IY05].

Los otros dos componentes del SCBRD forman el *Método de Razonamiento Difuso* (FRM, del inglés Fuzzy Reasoning Method) encargado de inferir la clase a la que pertenece un ejemplo.

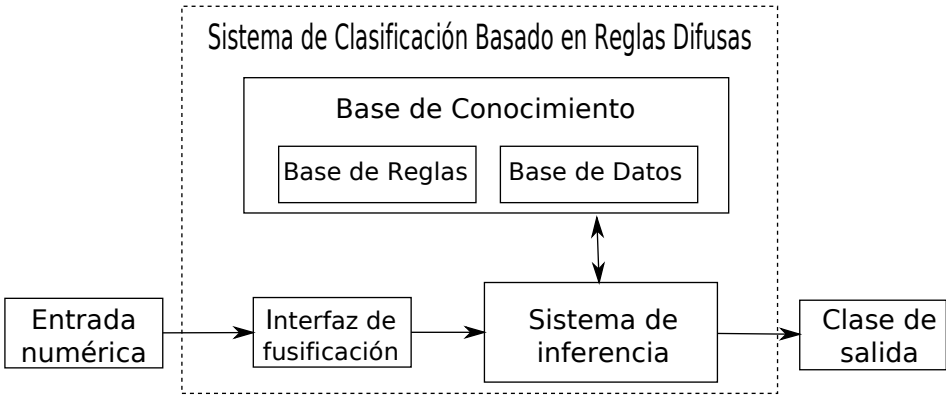


Figura 1: Estructura de un SCBRD de tipo Mamdani.

Cuando el sistema recibe un nuevo ejemplo, la interfaz de fusificación transforma los valores numéricos de entrada en valores difusos. Una vez realizada esta conversión, el sistema de inferencia emplea la información almacenada en la KB para predecir la clase. En esta memoria vamos a utilizar como base el siguiente FRM [CdJH99]. Sea $x = (x_1, \ldots, x_F)$ un nuevo ejemplo a clasificar, M el número de clases del problema y L el número de reglas que componen la RB. Los pasos del sistema de inferencia son:

1. *Grado de emparejamiento.* Calcular la *fuerza de activación del antecedente de todas las reglas con el ejemplo* x aplicando un operador de conjunción (normalmente una T-norma [Web83]).

$$\mu_{A_j}(x) = T\left(\mu_{A_{j1}}(x_1), \ldots, \mu_{A_{jF}}(x_F)\right), \qquad j = 1, \ldots, L. \tag{I.1}$$

2. *Grado de asociación.* Ponderar el grado de emparejamiento de cada regla empleando el peso de la regla.

$$b_j(x) = \mu_{A_j}(x) \cdot RW_j \tag{I.2}$$

3. *Clasificación.* Clasificar el ejemplo en base a los grados de asociación. Los dos métodos más comunes [CdJH99] son:

   - *Regla ganadora* (en inglés *winning rule*). Se predice la clase de la regla con el mayor grado de asociación.

$$clase = arg \max_{m=1,\ldots,M} \left(\max_{R_j \in RB;\, C_j = m} b_j(x) \right) \tag{I.3}$$

   - *Combinación aditiva* (en inglés *Additive combination*). Para cada clase, se calcula la suma de los grados de asociación correspondientes a la clase, obteniendo su *confianza*. Finalmente, se predice la clase que tiene mayor confianza.

$$\begin{aligned} conf_m(x) &= \sum_{R_j \in RB;\, C_j = m} b_j(x), \;\; m = 1, \ldots, M \\ clase(x) &= arg \max_{m=1,\ldots,M} (conf_m(x)) \end{aligned} \tag{I.4}$$

En esta memoria nos centramos en dos conocidos SCBRDs: el algoritmo de Chi et al. [CYP96] y FARC-HD [AFAH11]. Dado que ambos métodos se basan en la misma estructura de regla y emplean el mismo FRM, en las siguientes subsecciones describimos únicamente el correspondiente proceso de aprendizaje.

2.2.1. Algoritmo de Chi et al.

El proceso de aprendizaje de Chi et al. [CYP96] consiste en los siguientes pasos:

1. *Construcción de las etiquetas lingüísticas.* Para cada variable se construyen los correspondientes conjuntos difusos empleando la misma función de pertenencia triangular y se distribuyen de manera uniforme a lo largo del rango de valores de la variable.
2. *Generación de una regla para cada ejemplo.* Para cada ejemplo de entrenamiento x se genera una nueva regla difusa de la siguiente forma.
   a) Se calcula el grado de pertenencia de cada valor x_f a todos los conjuntos difusos asociados a la variable f.
   b) Para cada variable, se selecciona la etiqueta lingüística con el mayor grado de pertenencia.
   c) Se determina la parte antecedente de la regla mediante la intersección de las etiquetas lingüísticas seleccionadas. La parte consecuente es la propia etiqueta del ejemplo (y). Todas las reglas tienen el mismo número de antecedentes, es decir, tantos antecedentes como variables haya en el problema (F).
   d) Se calcula el peso de la regla, normalmente mediante el factor de certeza [IY05]:

$$RW_j = \frac{\sum_{x_i \in TR_{C_j}} \mu_{A_j}(x_i)}{\sum_{i=1}^{N} \mu_{A_j}(x_i)}, \tag{I.5}$$

   donde TR_{C_j} es el conjunto de ejemplos de entrenamiento pertenecientes a la clase de la regla (C_j), N es el número de ejemplos de entrenamiento, y $\mu_{A_j}(x_i)$ es el grado de emparejamiento de la regla definido en la Ecuación I.1.

Como se puede apreciar, podrían obtenerse reglas que compartan la parte antecedente y que difieran en el consecuente. A esta situación se le denomina *conflicto*, y se resuelve escogiendo la regla con mayor peso.

2.2.2. FARC-HD

FARC-HD (Fuzzy Association Rule-Based Classification Model for High-Dimensional Problems) [AFAH11] es uno de los SCBRDs más precisos e interpretables de la literatura. El algoritmo de aprendizaje está compuesto por tres fases:

1. *Extracción inicial de reglas difusas asociativas*: En esta primera fase se emplea un árbol de búsqueda [AS94] para cada clase con el objetivo de generar una base de reglas preliminar compuesta por reglas difusas asociativas. Para ello, se extraen los *itemsets* más frecuentes empleando la confianza y el soporte, donde un itemset es un conjunto de etiquetas lingüísticas. Finalmente, se generan las reglas difusas a partir de los itemsets más frecuentes. La longitud máxima de las reglas viene dada por la profundidad máxima del árbol de búsqueda.

2. *Selección de reglas candidatas*: Para seleccionar las reglas más interesantes entre las obtenidas en la etapa anterior se emplea un algoritmo de *subgroup discovery* mediante un esquema de ponderación [KL06]. Esta ponderación está basada en el grado de cubrimiento de las reglas.

3. *Algoritmo evolutivo para la selección de reglas y ajuste lateral*: Se emplea un algoritmo evolutivo para realizar un ajuste lateral de las etiquetas lingüísticas [AAFH07] y seleccionar las reglas más precisas.

Generalmente los modelos construidos por FARC-HD son más compactos y precisos que los de Chi. El promedio de número de reglas y antecedentes suele ser significativamente menor y las etiquetas lingüísticas están optimizadas y ajustadas a los datos. Estas propiedades permiten aumentar la capacidad de generalización y el nivel de interpretabilidad de los modelos.

2.3. Problemas multi-clase

Los problemas de clasificación multi-clase son aquellos en los que el clasificador debe discernir entre más de dos clases. Este tipo de escenario es habitual en campos como la clasificación de microarrays [LX09] y tejidos [TCM04], el reconocimiento de imágenes [KSH12, HZRS15] y del lenguaje de signos [AA10], o la clasificación de tipos de cáncer [AS09] y de señales de electroencefalogramas [GU07].

Generalmente la dificultad de construir clasificadores para problemas en los que solamente hay dos clases (binarios) es más sencillo que en el caso de los multi-clase. Ésto es debido a que la complejidad de las fronteras de decisión suele aumentar cuando se considera un número mayor de clases. Motivadas por este hecho surgen las estrategias de descomposición, las cuales afrontan los problemas multi-clase dividiéndolos en problemas binarios más sencillos de resolver. Cada uno de estos problemas binarios es abordado por un clasificador independiente denominado *clasificador base* que se especializa únicamente en un par de clases [FÖ2]. Cuando se clasifica un nuevo ejemplo, las predicciones de todos los clasificadores base se agregan y se obtiene la clase predicha. De acuerdo con varios estudios, este tipo de metodología ha sido capaz de mejorar el rendimiento de clasificadores que soportan problemas multi-clase de manera inherente [GFB+11, FÖ2, Für03]. Además, su utilización permite aplicar clasificadores binarios populares como las SVM [Vap98].

En la literatura especializada se han propuesto diferentes estrategias de descomposición [LCG08]. En esta tesis nos centramos en dos de las estrategias más populares: la estrategia uno-contra-uno (OVO, del inglés *One-vs-One*) [KPD90] (también conocida como aprendizaje por parejas o *pairwise learning*) y uno-contra-todos (OVA, del inglés *One-vs-All*) [CB91, AMMR95]. Para ilustrar

el funcionamiento de estas dos estrategias en las siguientes subsecciones, utilizaremos como referencia el problema multi-clase (de tres clases) mostrado en la Figura 2.

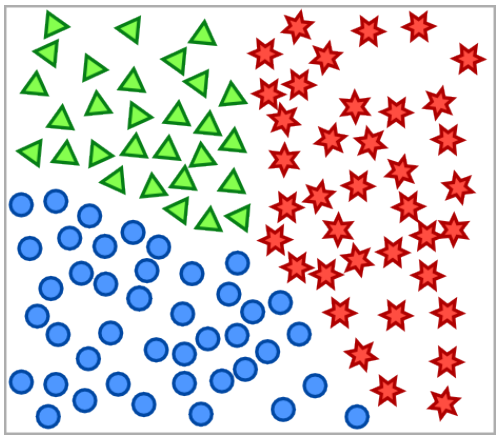


Figura 2: Problema multi-clase de tres clases y dos atributos.

2.3.1. Estrategia uno-contra-uno

La estrategia OVO divide un problema de M clases en $M(M-1)/2$ problemas binarios, tantos como posibles combinaciones de pares de clases (Figura 3). Cada uno de estos problemas es afrontado por un clasificador base que distingue entre el par de clases $\{C_i, C_j\}$. Cuando se clasifica un ejemplo, cada clasificador devuelve un par de grados de confianza $r_{ij}, r_{ji} \in [0,1]$ a favor de las clases C_i y C_j, respectivamente. Todas estas salidas (confianzas) se almacenan en una *matriz de votos* R:

$$R = \begin{pmatrix} - & r_{12} & \cdots & r_{1M} \\ r_{21} & - & \cdots & r_{2M} \\ \vdots & & & \vdots \\ r_{M1} & r_{M2} & \cdots & - \end{pmatrix} \tag{I.6}$$

Dado que cada sub-problema es abordado por un clasificador base independiente, suele ser recomendable normalizar la matriz de votos para que todas las confianzas estén en el mismo rango de valores. Este proceso de normalización es importante cuando se emplean clasificadores que no devuelven confianzas en el intervalo $[0,1]$ que puedan ser interpretadas como probabilidades, como es el caso de los SCBRDs. La normalización de la matriz se realiza de la siguiente forma:

$$\hat{r}_{ij} = \begin{cases} \dfrac{r_{ij}}{r_{ij} + r_{ji}} & \text{if } r_{ij} \neq 0 \text{ or } r_{ji} \neq 0 \\ 0{,}5 & \text{if } r_{ij} = r_{ji} = 0 \end{cases} \tag{I.7}$$

Una vez se han obtenido y normalizado las salidas de todos los clasificadores, el ejemplo se clasifica agregando todos estos valores mediante un método de agregación. En esta memoria consideramos los siguientes métodos:

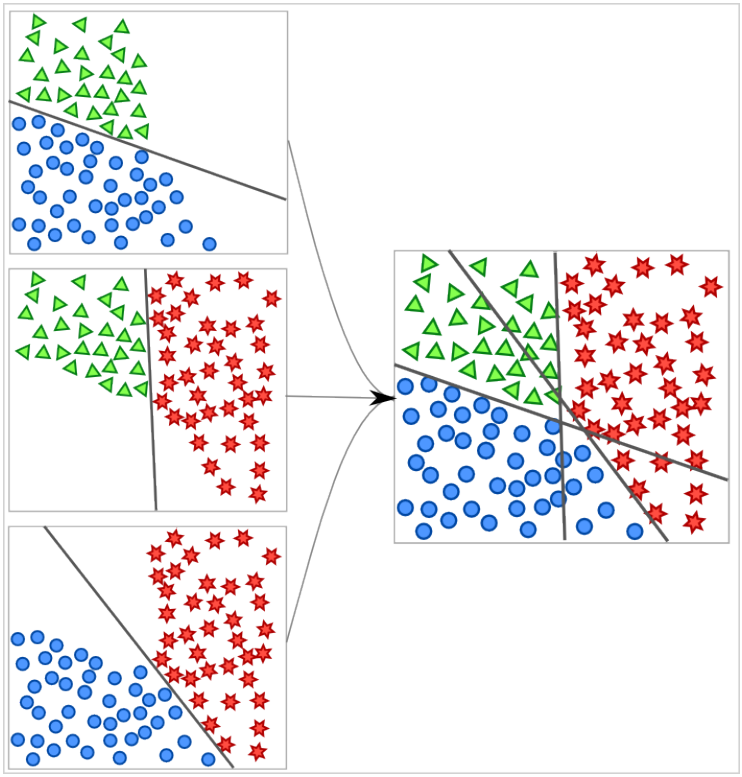


Figura 3: Problema multi-clase descompuesto mediante OVO.

- *Voto simple* [Fri96]. Cada clasificador base devuelve un voto para la clase predicha en su problema. Finalmente, se predice la clase con el mayor número de votos:

$$Clase = arg \max_{i=1,\dots,M} \sum_{1 \le j \ne i \le M} s_{ij} \tag{I.8}$$

donde s_{ij} es 1 si $\hat{r}_{ij} > \hat{r}_{ji}$ y 0 en caso contrario.

- *Voto ponderado* [HV10]. Cada clasificador base devuelve un par de grados de confianza asociados a las dos clases. Se predice la clase que obtiene la mayor confianza acumulada:

$$Clase = arg \max_{i=1,\dots,M} \sum_{1 \le j \ne i \le M} \hat{r}_{ij} \tag{I.9}$$

- *Criterio de no-dominancia* [FCB$^+$10]. La matriz de votos es considerada como una relación de preferencia difusa. Posteriormente se calcula el grado de no-dominancia y se predice la clase que obtiene el valor más alto:

$$Clase = arg \max_{i=1,\dots,M} \left\{ 1 - \max_{j=1,\dots,M} r'_{ji} \right\} \tag{I.10}$$

donde R' es la matriz de votos estricta (después de la normalización).

- *Learning valued preference for classification* (LVPC) [HH09b, HB08]. Al igual que en la estrategia del criterio de no-dominancia, la matriz de votos es considerada una relación de preferencia difusa. En base al modelado de la preferencia difusa, la relación original se descompone en tres nuevas relaciones con significados diferentes: preferencia estricta, conflicto, e ignorancia. Finalmente, se establece una regla de decisión basada en la estrategia del voto:

$$Clase = arg \max_{i=1,\dots,M} \sum_{1 \le j \ne i \le M} P_{ij} + \frac{1}{2} C_{ij} + \frac{N_i}{N_i + N_j} I_{ij} \tag{I.11}$$

siendo N_i el número de ejemplos de entrenamiento pertenecientes a la clase i, C_{ij} el grado de conflicto (grado en el que las dos clases están soportadas), I_{ij} el grado de ignorancia (grado en el que ninguna de las clases está soportada), y P_{ij} y P_{ji} la relación estricta para i y j, respectivamente. Estas variables se calculan de la siguiente manera:

$$C_{ij} = \text{mín}\left\{\hat{r}_{ij}, \hat{r}_{ji}\right\}$$
$$P_{ij} = \hat{r}_{ij} - C_{ij}$$
$$P_{ji} = \hat{r}_{ji} - C_{ij}$$
$$I_{ij} = 1 - \text{máx}\left\{\hat{r}_{ij}, \hat{r}_{ji}\right\}$$

2.3.2. Estrategia uno-contra-todos

En OVA un problema de M clases se divide en M problemas binarios, de tal forma que cada clasificador base se especializa en distinguir una de las clases del resto. La clase en la que se especializa se considera la *clase positiva*, mientras que el resto se etiquetan como la *clase negativa*. Cuando se clasifica un nuevo ejemplo, todos los clasificadores devuelven un grado de confianza $r_i \in [0, 1]$ asociado a la clase C_i en la que se han especializado. En esta caso, las salidas se almacenan en un *vector de votos* R:

$$R = (r_1, \ldots, r_i, \ldots, r_m) \tag{I.12}$$

Al igual que en OVO, el rango de valores devueltos por cada clasificador depende del sub-problema correspondiente. Estas diferencias en los rangos podrían llevar a clasificar el ejemplo incorrectamente, debido a que la comparación entre las confianzas podría no ser justa. Por tanto, el vector de votos R suele normalizarse para ajustar todos los valores al mismo rango. Este proceso de normalización se realiza respecto a la confianza asociada a la clase negativa de cada clasificador:

$$\hat{r}_i = \frac{r_i}{r_i + \overline{r}_i} \tag{I.13}$$

Finalmente, todos estos valores se agregan para predecir la clase del ejemplo. El método de agregación que se emplea habitualmente en OVA es el máximo, prediciendo la clase que obtiene la mayor confianza. A pesar de que existen alternativas como el *OVA ordenado dinámicamente* [HMCC08], en general no hay diferencias estadísticamente significativas respecto al máximo, siendo este último un método más simple.

2.4. Problemas de clasificación en Big Data

En en un escenario en el que la cantidad de información digital disponible se duplica cada dos años [GR12], una gran parte de los algoritmos de procesamiento de datos que venían utilizándose hasta la fecha han comenzado a mostrar claras limitaciones de escalabilidad. A esta problemática se le conoce como *Big Data*. Un problema Big Data es en general aquel en el que la cantidad de información a procesar excede la capacidad de cómputo o almacenamiento de un ordenador convencional moderno [Art13, GH15, NMS+15]. Entre las metodologías más populares para lidiar con este tipo de entornos se encuentra la *computación distribuida* [GGL03, DG08]. Las técnicas distribuidas dividen el conjunto de datos original en varios subconjuntos que se distribuyen a través de varios *nodos*. Un nodo puede ser tanto un PC de sobremesa convencional como un servidor especializado que se conecta a una red (generalmente local) formada por más nodos a la que llamamos *cluster*. Cuando se procesa un conjunto de datos, cada uno de estos nodos procesa únicamente el subconjunto de datos que le ha sido asignado. Finalmente, todos los resultados parciales generados en los nodos se agregan y se obtiene el resultado final.

Los algoritmos de aprendizaje automático no son una excepción a esta nueva problemática. Muchas de las técnicas que han venido destacando en tareas de clasificación no se pueden aplicar de forma secuencial en problemas Big Data. Generalmente esta limitación es debida a dos factores:

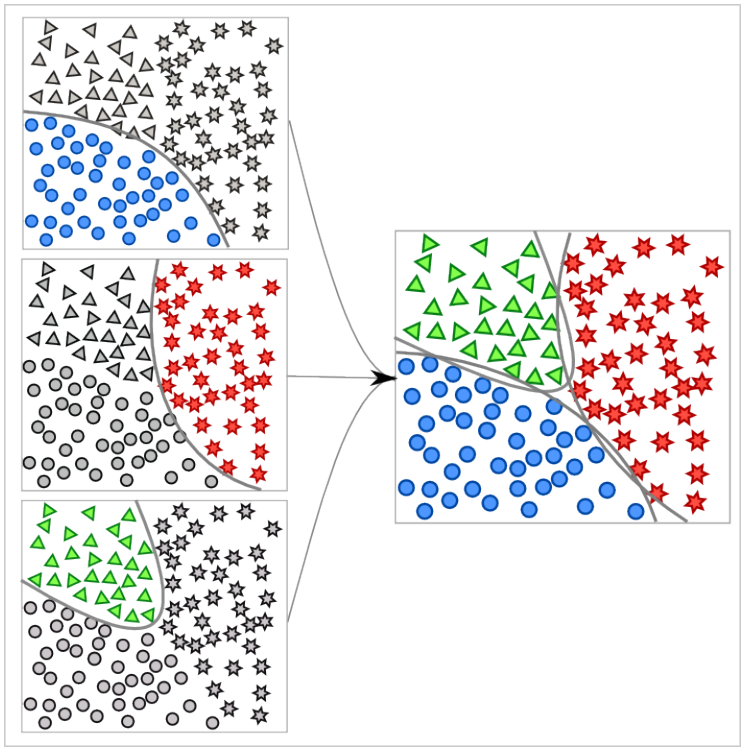


Figura 4: Problema multi-clase descompuesto mediante OVA.

- El coste computacional del algoritmo respecto al número de ejemplos o características del problema es en el mejor de los casos cuadrático. Por consiguiente, el aumento de tamaño del conjunto de datos afecta drásticamente al tiempo requerido para completar la tarea, siendo en muchos casos inaceptable.

- El conjunto de datos es demasiado grande como para poder cargarlo en la memoria principal. Dado que la mayoría de algoritmos de aprendizaje automático requieren que todos los ejemplos estén accesibles desde la memoria principal, en muchas ocasiones el algoritmo no puede ser siquiera ejecutado, teniendo en cuenta que la memoria RAM de un ordenador convencional es a día de hoy de unos 8GB.

Con el objetivo de poder superar estas limitaciones, los investigadores han comenzado a diseñar nuevas soluciones distribuidas para poder aplicar algoritmos de clasificación en grandes conjuntos de datos. La aparición de herramientas y *frameworks* de código abierto que proporcionan sistemas distribuidos transparentes al usuario ha sido clave en esta evolución. En esta tesis nos centramos en dos de las más populares: Apache Hadoop y Apache Spark.

Otro tipo de metodología que se ha empleado frecuentemente para aliviar los requerimientos computacionales y de almacenamiento de los algoritmos de aprendizaje automático son las técnicas de reducción de prototipos (PR) [NL11]. Los métodos de PR permiten que clasificadores que no están diseñados para Big Data puedan ejecutarse en estos entornos empleando una versión reducida de los datos. Además, diversos estudios han mostrado que este proceso de reducción ayuda a mejorar la precisión de estos algoritmos incluso en problemas que no son Big Data [GLH14].

En esta sección repasamos los conceptos básicos de Hadoop y Spark (Secciones 2.4.1 y 2.4.2) y presentamos el estado actual de los SCBRDs (Sección 2.4.3) y de las técnicas de PR en problemas Big Data (Sección 2.4.4).

2.4.1. Apache Hadoop

Apache Hadoop[1] es un framework de código abierto desarrollado por la fundación Apache que proporciona herramientas para la gestión y procesamiento de datos en sistemas distribuidos. Está basado en dos trabajos de Google: el *Google File System* (GFS) [GGL03] y el algoritmo *MapReduce* [DG08]. El núcleo de Hadoop consiste en un sistema de ficheros distribuido basado en GFS llamado HDFS y en una implementación libre del algoritmo MapReduce:

- *HDFS* (capa de almacenamiento): los ficheros se dividen en bloques físicos que son distribuidos entre todos los nodos. Cuando se procesa un fichero, el código del algoritmo se transfiere a los nodos para aprovechar la localidad de los datos. De esta forma, un nodo procesará mayormente bloques almacenados en sus discos duros, evitando la congestión de la red.

[1] `http://hadoop.apache.org`

- *MapReduce* (capa de procesamiento): es el paradigma empleado para procesar los datos del HDFS. Está compuesto por dos fases: la fase *Map* y la fase *Reduce*. En la fase Map cada nodo procesa un sobconjunto de los datos y genera una serie de resultados parciales. Estos resultados se agregan posteriormente en la fase Reduce para generar el resultado final. A continuación se detalla el funcionamiento de estas dos fases:

  1. *Fase Map*: los datos de entrada se dividen en particiones lógicas asociadas a diferentes bloques físicos alojados en HDFS. Cada una de estas particiones es procesada por una única unidad de procesamiento llamada *mapper*. Un mismo nodo puede ejecutar múltiples mappers al mismo tiempo. A la hora de procesar una partición, los datos de entrada se transforman en un conjunto de pares (k, v) que son procesados por la función *map()* (definida por el usuario). Esta función es invocada para cada par (k, v), devolviendo un nuevo par (k', v') que formará parte de los denominados *datos intermedios*. Finalmente, estos datos intermedios son transferidos a los *reducers*, encargados de ejecutar la fase Reduce, siguiendo los siguientes pasos:
     a) Sorting y Merging: todos los pares generados en los mappers son ordenados por clave y todos los valores asociados a la misma clave son agrupados en una lista de valores (k', $list(v')$).
     b) Partitioning: a cada clave se le asigna un reducer.
     c) Shuffle: los pares ($k', list(v')$) son transferidos a los reducers.

     Además de la función *map()*, existe una función llamada *cleanup()* que es invocada una sola vez al final de la ejecución de cada mapper, una vez los datos de entrada han sido procesados por la función *map()*. La función *cleanup()* puede devolver también una serie de pares (clave, valor) que formarán parte de los datos intermedios. Generalmente, solamente una de las dos funciones será la responsable de generar los datos intermedios.
  2. *Fase Reduce*: conforme los mappers van terminando de procesar los datos, los reducers ordenan y agrupan los pares generados en los mappers por clave. Posteriormente, una vez ha finalizado la ejecución de los mappers, los valores asociados a cada k' ($list(v')$) se agregan en la función *reduce()* (definida por el usuario), generándose el resultado final para esa clave v''.

  La Figura 5 describe el flujo de datos del paradigma MapReduce. Además de las fases Map y Reduce, existe una fase extra destinada a optimizar la ejecución de MapReduce llamada *Combine*. Esta fase se ejecuta de forma local en cada salida de la fase Map ($k', list(v')$) y se emplea como un mini-reducer para reducir la cantidad de datos intermedios que se transfieren a los reducers. De hecho, en muchas ocasiones el código del combiner es prácticamente idéntico al del reducer. Todas estas fases (Map + Combine + Reduce) componen un trabajo (*job*) MapReduce.

2.4.2. Apache Spark

Apache Spark[2] fue presentado como una generalización y extensión del paradigma MapReduce,

[2] `https://spark.apache.org`

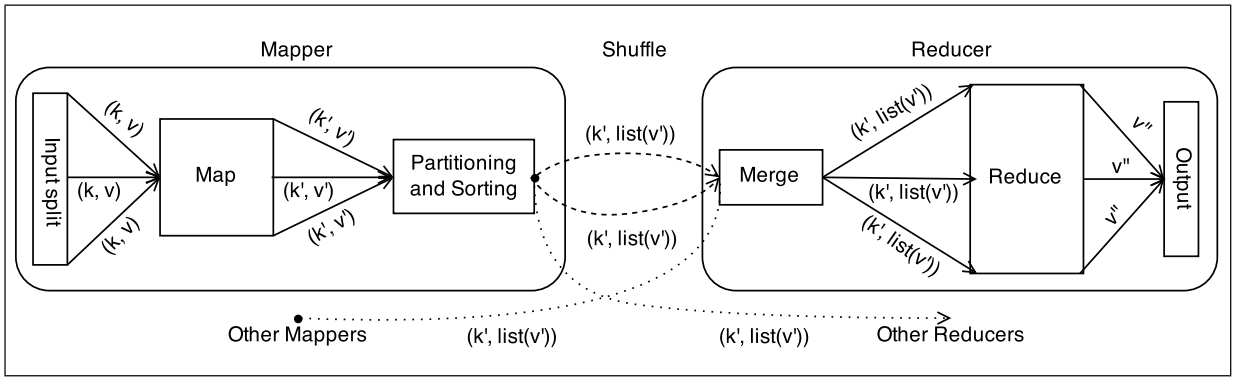


Figura 5: Flujo de datos del paradigma MapReduce.

soportando funciones de *caché* que minimizan el sobre-coste de las operaciones de entrada/salida en disco. Las funciones de caché permiten almacenar los datos intermedios en la memoria principal, evitando que se almacenen en el disco duro y acelerando considerablemente la ejecución de los algoritmos que reutilizan estos datos. Spark se construye alrededor del concepto de los *Resilient Distributed Datasets* (RDDs), los cuales representan conjuntos de datos distribuidos inmutables y operaciones (*transformaciones*) que son evaluadas de forma *perezosa*. Una estrategia de evaluación perezosa es aquella que retrasa el cálculo de una expresión hasta que su valor es necesario y que reutiliza dicho valor para evitar volver a evaluar la expresión. La ejecución de un algoritmo en Spark consiste en una secuencia de fases (en inglés *stages*) formadas por una serie de transformaciones que se descomponen a su vez en tareas (en inglés *tasks*). Una fase está compuesta únicamente por transformaciones que no implican un proceso de shuffling. Las tareas correspondientes a estas transformaciones son ejecutadas por los denominados *executors*, cada uno de los cuales representa un proceso independiente en la máquina virtual de Java (JVM) de un nodo determinado (*worker node*). El resultado de todas las transformaciones se calcula cuando se invoca una *acción*, como por ejemplo una función *reduce*. La Figura 6 ilustra el flujo de datos de todo este proceso.

Además de ser más rápido que Hadoop, Spark permite lanzar un número indefinido de trabajos MapReduce dentro del mismo programa, soportando una mayor variedad de algoritmos de procesamiento de datos que Hadoop.

2.4.3. Clasificadores difusos en Big Data

A pesar del gran rendimiento mostrado por los clasificadores difusos fuera del ámbito del Big Data, mantener las propiedades que han popularizado estas técnicas cuando se abordan grandes conjuntos de datos ha resultado ser un gran reto. En los últimos tres años, apenas se han propuesto una docena de métodos difusos para problemas de clasificación Big Data [DMS15,EGSB17, FdRH16,FAH17,FMS$^+$17,GGGP16,GVGdJC17,LdRBH15,PRRRPG$^+$17,RZGP17,SMP17,SBDM17].

En general, todas estas aproximaciones están basadas o bien en el aprendizaje incremental [GGGP16,RZGP17] o bien en la computación distribuida [DMS15,EGSB17,FdRH16,FAH17,

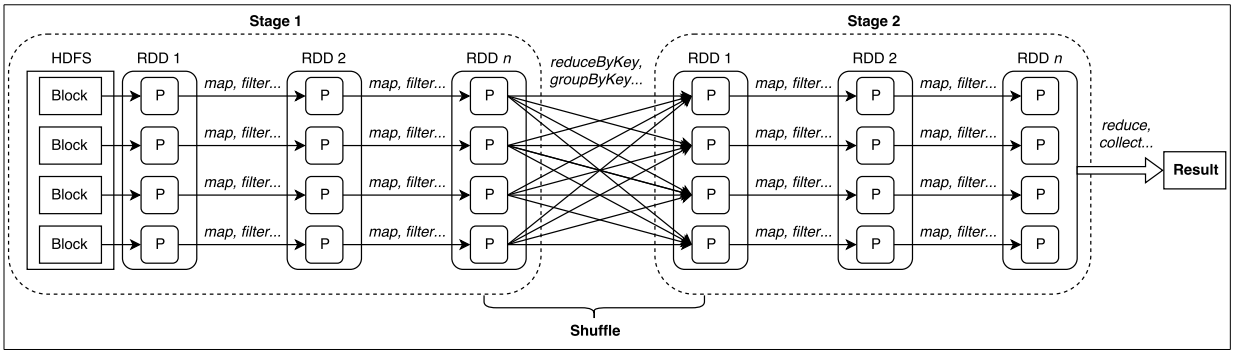


Figura 6: Flujo de datos en Spark (P = Partición).

FMS$^+$17, GVGdJC17, LdRBH15, PRRRPG$^+$17, SMP17, SBDM17]. En el primer caso, el conjunto de entrenamiento se divide en una serie de subconjuntos denominados *episodios* que forman los datos de entrada del sistema. Posteriormente, el clasificador aprende de cada episodio de manera secuencial incorporando el conocimiento adquirido en el episodio anterior. Una de las ventajas de esta metodología es que no requiere un cluster de computación para ejecutar el algoritmo, ya que el problema inicial se descompone en múltiples problemas de clasificación secuenciales que son equivalentes a un problema *no Big Data* en términos de complejidad computacional. Sin embargo, esta característica hace que el conocimiento que surge de las inter-relaciones entre los diferentes episodios pueda perderse, ya que el clasificador no dispone de una visión global del conjunto de entrenamiento. En cuanto a la computación distribuida, la fase de aprendizaje se realiza siguiendo los paradigmas mostrados en las dos secciones anteriores. La principal diferencia respecto al aprendizaje incremental es que el proceso de aprendizaje se realiza de forma concurrente empleando el conjunto de entrenamiento al completo (si bien algunos métodos se basan en la ejecución de múltiples optimizaciones locales concurrentes). Por tanto, a pesar de que pueda requerir mayores prestaciones de *hardware*, la computación distribuida soluciona el problema del aprendizaje incremental asociado a la falta de visión de conjunto. En esta tesis nos centramos únicamente en técnicas distribuidas.

Los métodos difusos distribuidos pueden a su vez afrontar el problema de clasificación o bien descomponiendo el problema original en varios sub-problemas locales [LdRBH15, FdRH16, FAH17] o bien realizando un proceso de aprendizaje global [EGSB17, FMS$^+$17, GVGdJC17, SMP17, SBDM17], o incluso combinado estas dos aproximaciones [DMS15, PRRRPG$^+$17]. En el primer caso, cada sub-problema es afrontado por un clasificador independiente (pudiendo ser un clasificador no distribuido existente) y el modelo final es construido agregando todos los modelos locales obtenidos en los diferentes sub-problemas. De forma similar al aprendizaje incremental, este tipo de aprendizaje es vulnerable a la distribución y al tamaño de los subconjuntos y puede perder información que está disponible únicamente cuando se trata el conjunto de entrenamiento de forma global. En el caso de los modelos de aprendizaje global, la principal dificultad radica en la complejidad del diseño de algoritmos de aprendizaje paralelizables.

Las diferentes aproximaciones difusas presentadas hasta la fecha incluyen una gran variedad de técnicas de clasificación, como árboles de decisión (FDTs, del inglés Fuzzy Decision

Trees) [FMS+17, SMP17], sub-group discovery (SD) [PRRRPG+17], clasificadores asociativos (FACs, del inglés Fuzzy Associative Classifiers) [DMS15,SBDM17], minería de patrones emergentes (EPM, del inglés Emerging Patterns Mining) [GVGdJC17], y SCBRDs [EGSB17, FdRH16, FAH17, FMS+17, LdRBH15]. Sin embargo, todos estos trabajos presentan una limitación en común: la mayoría de ellos no es capaz de mantener la interpretabilidad del modelo y la precisión al mismo tiempo. Algunos algoritmos se centran en la precisión y generan modelos formados por demasiadas reglas [EGSB17,FdRH16,LdRBH15], reglas excesivamente largas [EGSB17,FdRH16, FAH17,LdRBH15], o un número demasiado elevado de etiquetas lingüísticas [SMP17,SBDM17]. Por otro lado, generalmente las metodologías que se centran en optimizar la interpretabilidad del modelo no son capaces de conseguir resultados del estado-del-arte en términos de precisión [FMS+17]. Además, existen contribuciones interesantes que, bajo nuestro punto de vista, no han desarrollado un estudio experimental suficientemente completo como para evaluar su eficiencia en entornos Big Data [DMS15, GVGdJC17, PRRRPG+17].

2.4.4. Reducción de prototipos en Big Data

Las técnicas de reducción de prototipos (PR) [GDCH12,NL11] corresponden a la categoría de los métodos de reducción de instancias, los cuales reciben el nombre de reducción de conjuntos de entrenamiento (en inglés, *training set reduction*) o reducción de prototipos, dependiendo del clasificador objetivo. Los métodos de PR se centran en los clasificadores basados en instancias, como por ejemplo el conocido *k-Nearest Neighbors* (k-NN) [Mcl04].

Los problemas de PR se pueden formular empleando la siguiente notación. Sea TR un conjunto de datos que contiene N ejemplos. Un problema de PR consiste en encontrar el mínimo subconjunto S de TR formado por N_S ejemplos (prototipos), con $N_S < N$, que maximicen la precisión en la clasificación de ejemplos desconocidos [KG15]. En la literatura especializada se han propuesto dos tipos de aproximaciones: la selección de prototipos (PS) [GDCH12], que trata de seleccionar los ejemplos más representativos del conjunto TR, y la generación de prototipos (PG) [TDGH12], que genera nuevos ejemplos a partir de los ejemplos existentes en TR. Existen también algunos métodos que combinan estas dos aproximaciones [TGH11].

A pesar del buen rendimiento mostrado por estas técnicas en términos de precisión y reducción, su aplicabilidad en problemas Big Data se ve seriamente afectada por las limitaciones de escalabilidad de la mayoría de las propuestas. El trabajo de Cano et al. [CHL05] fue uno de los primeros en abordar esta problemática. Los autores presentaron un método de estratificación que divide el conjunto de entrenamiento en múltiples subconjuntos manteniendo la distribución de las clases. Posteriormente, en cada subconjunto se aplica un método de PS existente y se agregan los ejemplos seleccionados entre todos los subconjuntos. Considerando un algoritmo de PS con una complejidad computacional cuadrática $O(N^2)$ y tomando T subconjuntos, la complejidad final es de $O(N^2/T^2)$ cuando se ejecuta en paralelo y de $O(N^2/T)$ en el caso secuencial, obteniendo una aceleración de T^2 y T, respectivamente. Siguiendo con esta metodología, varios autores propusieron diferentes métodos en una serie de trabajos [CHL07, GCH08, TGH11, TPB+15]. Sin embargo, todos ellos heredaban gran parte de los defectos presentes en la contribución de Cano et al. Por una parte, un aumento en el grado de paralelismo (número de subconjuntos) puede provocar

pérdidas importantes en la precisión del método. Este comportamiento negativo viene dado por el hecho de que en cada subconjunto se aplica un proceso de optimización local sin considerar el resto del conjunto de entrenamiento. Por consiguiente, es probable que el error de aproximación cometido en cada uno de estos subconjuntos sea cada vez mayor conforme el tamaño de los subconjuntos disminuya. Por otra parte, el proceso de estratificación requiere que el conjunto de entrenamiento sea almacenado en la memoria principal, limitando su aplicabilidad en conjuntos de datos de tamaño arbitrario. Finalmente, la combinación directa de los ejemplos seleccionados puede resultar en ejemplos ruidosos y/o redundantes.

Con el objetivo de plantear una solución a estas cuestiones, Triguero et al. presentaron una aproximación basada en MapReduce (MRPR) [TPB+15] que aplica una estrategia de *divide y vencerás* para emplear técnicas de PR existentes en Big Data. En este método, cada mapper ejecuta un proceso de reducción local en la correspondiente partición utilizando un determinado algoritmo de PR. Posteriormente, los ejemplos obtenidos en los mappers se agregan empleando diferentes técnicas que eliminan los ejemplos ruidosos y redundantes. A pesar de solucionar la mayoría de problemas de la estratificación, esta metodología sigue siendo vulnerable al incremento del grado de paralelismo.

A pesar de que se han propuesto nuevas aproximaciones basadas en la estratificación que resuelven gran parte de sus defectos [dHGGPdC12], la mayoría de ellas se centran en aplicar técnicas de PS o PG existentes en múltiples problemas locales para después agregar todas estas soluciones localmente óptimas. Los dos grandes inconvenientes de esta metodología son:

- El grado de paralelismo afecta al rendimiento. El hecho de que no se realice una optimización global considerando el conjunto de entrenamiento al completo hace que el método pierda efectividad conforme el tamaño de los subconjuntos disminuye.
- La complejidad computacional cuadrática $O(N^2)$ que caracteriza a la mayoría de métodos de PR es heredado. Por consiguiente, si el incremento de tamaño del conjunto de datos es considerablemente mayor que el aumento en el número de procesadores, la aceleración obtenida por medio de una ejecución paralela se verá drásticamente reducida.

3. Motivación

Una vez presentados los principales conceptos sobre los que está centrada esta memoria, nos planteamos los siguientes problemas abiertos que motivan el presente proyecto de tesis:

- El uso de estrategias de descomposición ha permitido mejorar el rendimiento de los SCBRDs en problemas de clasificación multi-clase [IYN05,HB08,HH09a,FCB+10,SH11]. No obstante, la eficacia de estas técnicas depende de las confianzas (salidas) de los clasificadores base. Cuando los SCBRDs afrontan un problema multi-clase directamente, el valor numérico de las confianzas no se emplea más allá de la clasificación, y por tanto carece de importancia siempre que el orden de las confianzas para todas las clases se mantenga (se predice aquella con la mayor confianza). Sin embargo, cuando se emplean estrategias de descomposición,

los valores numéricos de las confianzas son clave para la fase de agregación y la posterior clasificación. El hecho de que existan varios tipos de SCBRDs con diferentes métodos de inferencia y de construcción de reglas nos hace pensar que las confianzas devueltas en cada caso pueden diferir notablemente, provocando que el comportamiento de las técnicas de descomposición varíe dependiendo del SCBRD. Sin embargo, todavía no se ha presentado ningún trabajo que muestre el efecto de la inferencia y de la construcción de reglas en la fase de agregación de las técnicas de descomposición.

- Como hemos visto en la Sección 2.2.1, el algoritmo de aprendizaje de Chi et al. [CYP96] consiste en aplicar una serie de pasos muy sencillos en donde cada ejemplo de entrenamiento genera una nueva regla sin considerar el resto de ejemplos. Esta característica hace que el método sea fácilmente paralelizable y adaptable al paradigma MapReduce. La primera contribución que adaptó el algoritmo de Chi et al. a MapReduce fue Chi-FRBCS-BigDataCS [LdRBH15]. En este trabajo los autores propusieron un proceso de aprendizaje distribuido basado en dos fases:

  1. *Generación de múltiples bases de reglas*: cada mapper entrena un clasificador Chi de forma local empleando únicamente los ejemplos contenidos en la correspondiente partición. Por consiguiente, los pesos de las reglas generadas en los mappers están calculados en base a un subconjunto del conjunto original de entrenamiento.
  2. *Agregación de la bases de regla*: todas las reglas generadas en la fase anterior se agregan directamente para obtener la base de reglas final.

  La principal limitación de esta metodología es que la precisión del método se ve afectada a medida que el número de mappers aumenta. Este comportamiento es consecuencia de un proceso de aprendizaje basado en optimizaciones locales que no consideran en ningún momento el conjunto de entrenamiento al completo. El hecho de que los pesos de las reglas se calculen empleando un subconjunto de los datos hace que la calidad de estos pesos dependa del tamaño y distribución de los subconjuntos. Dado que los pesos de las reglas son un factor clave en el proceso de inferencia de un SCBRD [IY05], el error de aproximación cometido en el cálculo de los pesos explica la bajada de rendimiento del método cuando se añaden mappers. A pesar de la idoneidad de Chi para MapReduce, todavía no se ha propuesto una solución distribuida para recuperar este algoritmo en problemas Big Data.

- Además de Chi, se han propuesto más tipos de clasificadores difusos distribuidos para problemas de clasificación Big Data [DMS15,FdRH16,FAH17,FMS$^+$17,GVGdJC17,PRRRPG$^+$17, SMP17, SBDM17]. Sin embargo, ninguno de ellos es capaz de conseguir resultados competitivos en términos de precisión e interpretabilidad al mismo tiempo. Los métodos que se centran en optimizar la precisión suelen obtener modelos demasiado complejos y pierden interpretabilidad [SMP17, SBDM17]. Otros consiguen generar modelos compactos pero no logran ser del todo competitivos en términos de precisión [FMS$^+$17]. Finalmente, existen varias contribuciones interesantes que, en nuestra opinión, no han incluido conjuntos de datos suficientemente grandes como para poder evaluar la escalabilidad y el rendimiento del método [DMS15, GVGdJC17, PRRRPG$^+$17].

- Las soluciones de PR planteadas hasta la fecha siguen teniendo problemas de escalabilidad cuando se aplican en grandes volúmenes de datos. La mayoría de las contribuciones se centran en aplicar estrategias de *divide y vencerás* para poder emplear técnicas de PR existentes de forma distribuida. Para ello, el conjunto de datos original se divide en una serie de subconjuntos que son reducidos de forma independiente y concurrente por un determinado algoritmo de PR. Posteriormente, todos los subconjuntos reducidos se combinan para obtener los prototipos. Este tipo de metodología presenta dos limitaciones importantes. Por un lado, el coste computacional cuadrático $O(N^2)$ que suele caracterizar a los métodos de PR es heredado cuando el aumento en el tamaño del conjunto de datos es mucho mayor que el incremento del grado de paralelismo (procesadores). Por otro lado, de forma similar al caso de la solución distribuida del SCBRD Chi (Chi-FRBCS-BigDataCS), el rendimiento de este método depende del tamaño y de la distribución de los subconjuntos. A día de hoy, todavía no se ha propuesto ninguna aproximación de PR con complejidad lineal $O(N)$ que realice un proceso de reducción global sobre el conjunto de entrenamiento al completo.

4. Objetivos

El objetivo principal de esta memoria es: *estudiar y mejorar el rendimiento de la sinergia de los SCBRDs y las estrategias de descomposición en problemas multi-clase* y *diseñar soluciones distribuidas para SCBRDs en problemas de clasificación Big Data*. La memoria está organizada en torno a cuatro grandes objetivos, uno por cada problema abierto expuesto en la sección anterior, que engloban al objetivo principal:

- *Estudiar y adaptar el método de razonamiento difuso de varios SCBRDs para mejorar su precisión en problemas multi-clase mediante las estrategias OVO y OVA*. Como punto de partida, nos hemos planteado analizar el comportamiento de OVO y OVA con uno de los SCBRDs más precisos e interpretables de la literatura: FARC-HD [AFAH11]. Hemos visto que las confianzas devueltas por este clasificador no eran adecuadas para su posterior agregación debido al uso del producto como operador de conjunción en la inferencia. Por ello nos hemos planteado diseñar un nuevo tipo de función de agregación que permite minimizar los efectos negativos del producto cuando se emplean estrategias de descomposición. Con el objetivo de generalizar esta solución a más tipos de clasificadores, hemos estudiado el efecto de los diferentes procesos de aprendizaje y métodos de inferencia de varios SCBRDs en el rendimiento de OVO y OVA. Para poder abarcar la mayor variedad de métodos de construcción y estructuras de reglas, consideramos cuatro representantes diferentes: Chi [CYP96], FARC-HD [AFAH11], SLAVE [GP99], y FURIA [HH09a].
- *Recuperar el modelo original de Chi et al. en problemas de clasificación Big Data*. Dadas las limitaciones mostradas por la anterior propuesta de Chi para Big Data (Chi-FRBCS-BigDataCS [LdRBH15]), nos planteamos recuperar el modelo original de Chi et al. [CYP96] en problemas Big Data aprovechando la idoneidad del proceso de generación de reglas para aplicar el paradigma de MapReduce. El objetivo es conseguir un método de aprendizaje global que utilice el conjunto de entrenamiento al completo para calcular los pesos de las reglas.

De esta forma el modelo obtenido será independiente del grado de paralelismo empleado para la ejecución del algoritmo.

- *Aprovechar la solución distribuida de Chi anterior para el diseño de un método de PR de complejidad lineal.* Una vez desarrollado una solución distribuida global para el método de Chi, nos planteamos utilizar las reglas extraídas por nuestro algoritmo para generar prototipos que permitan reducir el conjunto de entrenamiento original y mantengan la precisión del clasificador *k-Nearest Neighbors* (k-NN) [Mcl04]. El propósito es conseguir un método de PR de complejidad lineal que sea independiente del grado de paralelismo. Para ello, intentamos diseñar un proceso de reducción que utilice el conjunto de datos al completo y no introduzca ningún tipo de aproximación.

- *Desarrollar un modelo difuso compacto y preciso para problemas de clasificación Big Data.* Además de proponer una nueva solución distribuida para Chi, creemos que es importante conseguir mantener el nivel de interpretabilidad que han mostrado los SCBRDs en problemas *no Big Data*. Para lograrlo, necesitamos diseñar un algoritmo de aprendizaje distribuido más sofisticado que sea capaz de generar bases de reglas mucho más simples. El objetivo final es conseguir un modelo compacto que sea competitivo en términos de precisión.

5. Discusión de resultados

En esta sección presentamos una breve descripción de cada una de las publicaciones que componen la presente memoria. Además, incluimos los principales resultados obtenidos en su realización.

5.1. Mejorando la clasificación multi-clase en el clasificador difuso FARC-HD: sobre la sinergia de funciones de solapamiento n-dimensionales y estrategias de descomposición

En este trabajo proponemos aplicar las estrategias de descomposición OVO y OVA para mejorar la precisión de FARC-HD [AFAH11] en problemas de clasificación multi-clase. A la hora de desarrollar esta sinergia han surgido varias dificultades:

- El uso del producto como operador de conjunción en las reglas de FARC-HD hace que los grados de pertenencia bajos produzcan grados de emparejamiento que tienden demasiado rápido a 0. Como resultado, la variabilidad en las confianzas devueltas por este clasificador disminuye drásticamente cuando el método de razonamiento difuso (FRM) trabaja con valores (grados de pertenencia) pequeños. Este escenario no afecta al clasificador cuando se aplica en problemas multi-clase directamente, ya que la variabilidad en las confianzas no afecta a la clasificación, siempre que el orden de las confianzas de cada clase se mantenga. Sin embargo, cuando se considera la fase de agregación de las estrategias OVO y OVA, esta variación es clave para poder retener el conocimiento de los clasificadores base. Dado que en cada problema binario pueden obtenerse reglas con longitudes diferentes, el uso del producto

como operador de conjunción (Ecuación I.1) puede penalizar a aquellos clasificadores base que generen reglas más largas, reduciendo la variabilidad en sus predicciones.

- Algunos de los métodos de agregación que han mostrado un rendimiento robusto y preciso en OVO, como el voto ponderado (WV) [HV10, GFB$^+$11, GFB$^+$13], no obtienen tan buenos resultados cuando se emplea FARC-HD. Nuestra hipótesis es que las estimaciones de las confianzas de las clase no predichas realizadas por FARC-HD distorsionan la fase de agregación.

Con el objetivo de mejorar la combinación de FARC-HD y OVO/OVA, hemos propuesto las siguientes soluciones:

- Adaptar el FRM de FARC-HD para obtener confianzas más adecuadas para la agregación en OVO/OVA. Para ello, hemos reemplazado el producto por otro tipo de funciones de agregación denominadas *funciones de solapamiento*. Esta adaptación permite minimizar el efecto de saturación que tienen los grados de pertenencia bajos en las confianzas devueltas por FARC-HD, aumentando su variabilidad en los valores cercanos a 0. No obstante, dado que las funciones de solapamiento están definidas únicamente para dos dimensiones, hemos definido el concepto de *funciones de solapamiento n-dimensionales* para poder calcular el solapamiento entre los valores de un vector de dimensionalidad arbitraria. A pesar de que el producto es un caso específico de estas funciones, en esta memoria utilizamos el término de función de solapamiento para referirnos a todas las funciones de solapamiento diferentes al producto.
- Presentar un nuevo método de agregación para OVO llamado *WinWV* que soluciona los problemas de WV con las confianzas de las clases no predichas en FARC-HD.

Para analizar y evaluar el comportamiento de nuestra propuesta, hemos desarrollado un estudio empírico que aborda las siguientes cuestiones:

- Analizamos el efecto de las funciones de solapamiento en el comportamiento final del modelo y estudiamos los beneficios de WinWV respecto a WV.
- Mostramos la mejora obtenida por medio de la combinación FARC-HD + OVO/OVA respecto al clasificador FARC-HD original. Además, realizamos una comparativa entre los diferentes métodos de agregación OVO.
- Estudiamos la precisión obtenida por nuestra propuesta en comparación con otros clasificadores difusos del estado-del-arte como FURIA [HH09a], IVTURS [SFBH13], and PTTD [SH11].

Las conclusiones extraídas de este estudio son las siguientes:

- Hemos comprobado que el uso de estrategias de descomposición con FARC-HD es beneficioso para afrontar problemas multi-clase. Además, la modificación del FRM propuesta en este trabajo ha mostrado que este beneficio es aún mayor cuando se adapta la inferencia de este clasificador al proceso de agregación de OVO/OVA.

- Las funciones de solapamiento que mejores resultados obtienen son aquellas que mantienen la mayor variabilidad (menor saturación) en las confianzas y conservan la idempotencia.
- El nuevo método de agregación que hemos presentado para OVO (WinWV) ha resuelto los problemas de WV con las confianzas de las clases no predichas.
- La sinergia de FARC-HD y OVO consigue resultados competitivos respecto a los clasificadores difusos del estado-del-arte.

La publicación asociada a este trabajo es la siguiente:

- M. Elkano, M. Galar, J. Sanz, A. Fernández, E. Barrenechea, F. Herrera, H. Bustince, "Enhancing Multiclass Classification in FARC-HD Fuzzy Classifier: On the Synergy Between n-Dimensional Overlap Functions and Decomposition Strategies", *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 5, pp. 1562–1580, 2015.

5.2. Sistemas de clasificación basados en reglas difusas para problemas multiclase utilizando estrategias de descomposición binaria: sobre la influencia de las funciones de solapamiento n-dimensionales en el método de razonamiento difuso

A partir de las conclusiones extraídas del uso de estrategias de descomposición en FARC-HD, hemos llevado a cabo un amplio estudio del comportamiento de este tipo de sinergias en diferentes SCBRDs. Los objetivos de este trabajo son los siguientes:

- Estudiar el efecto que pueden tener diferentes tipos de métodos de aprendizaje, estructuras de regla, y FRMs en la fase de agregación.
- Analizar el rendimiento de las funciones de solapamiento n-dimensionales en varios SCRBDs y valorar si la solución presentada para FARC-HD es beneficiosa cuando se emplean otros clasificadores difusos. Además del comportamiento de las estrategias de descomposición, queremos evaluar también el rendimiento de las funciones de solapamiento en cada uno de los SCRBDs cuando se aplican de forma directa (sin considerar OVO y OVA).
- Comprobar si el método de agregación para OVO propuesto en el trabajo anterior (WinWV) mejora la precisión de WV en el resto de los SCBRDs considerados.

Para realizar este estudio empleamos cuatro representantes de diferentes variantes de SCBRDs: Chi [CYP96], FARC-HD [AFAH11], SLAVE [GP99], y FURIA [HH09a]. Las lecciones aprendidas en cada uno de los clasificadores son las siguientes:

- *Chi*
  - *Aplicación directa*: las funciones de solapamiento que mejoraban el rendimiento de FARC-HD perjudican la precisión de Chi. Esta diferencia de comportamiento radica

en que Chi siempre obtiene reglas compuestas por el mismo número de antecedentes (tantos como variables) que son construidas considerando todas las clases al mismo tiempo. Por consiguiente, las reglas de Chi están más próximas entre sí que las de FARC-HD y requieren una mayor capacidad discriminación. Las gráficas mostradas en el estudio revelan que las funciones de solapamiento *suavizan* las fronteras de decisión para valores que están en la mitad superior del rango de la variable. Esta propiedad es la que permite evitar la saturación en los extremos inferiores cuando se emplea FARC-HD.

- *OVO y OVA*: debido a la drástica bajada en precisión sufrida en los clasificadores base, las estrategias de agregación se ven seriamente afectadas. Por consiguiente, las funciones de solapamiento no son capaces de ofrecer el beneficio mostrado en el caso de FARC-HD.

- *SLAVE*
  - *Aplicación directa*: al igual que Chi, SLAVE genera reglas más específicas (con más antecedentes) que FARC-HD y requiere una capacidad de discriminación mayor que la ofrecida por las funciones de solapamiento. Sin embargo, la principal diferencia en la estructura de las reglas es el uso de antecedentes que están compuestos por conjuntos de etiquetas lingüísticas unidas por un operador de disyunción. En las reglas de FARC-HD un antecedente está formado por una única etiqueta lingüística y no se emplea ningún tipo de operador de disyunción. Creemos que esta diferencia en la estructura de las reglas distorsiona el efecto de las funciones de solapamiento.
  - *OVO y OVA*: a pesar de no haber encontrado diferencias estadísticas en el uso de diferentes funciones de solapamiento, hemos comprobado que el ratio de mejora respecto al producto es mayor que en el caso de la aplicación directa de SLAVE. Este hecho sugiere que las estrategias de descomposición se están beneficiando de las propiedades de las funciones de solapamiento, pero los clasificadores base no aportan suficiente capacidad de discriminación. Por otro lado, la estrategia OVA ha mostrado peores resultados que en el resto de SCBRDs. Este comportamiento se debe al problema del imbalanceo de clases inherente al modelo OVA y a la incapacidad de SLAVE para tratar con estas situaciones.

- *FURIA*: en el caso de FURIA, el rendimiento de las diferentes funciones de solapamiento en la aplicación directa y en las estrategias de descomposición es parecido. El hecho de que FURIA construya diferentes conjuntos difusos para cada regla permite que las funciones de pertenencia estén altamente ajustadas a los datos y devuelvan grados de pertenencia muy altos o nulos (0). Sin embargo, las funciones de solapamiento n-dimensionales fueron diseñadas para aumentar la variabilidad de las confianzas cuando se trabaja con valores (grados de pertenencia) cercanos a 0. Por consiguiente, los escenarios propuestos por FURIA y FARC-HD difieren notablemente, lo que explica que FURIA no pueda beneficiarse de las propiedades de las funciones de solapamiento.

Dado que el comportamiento de FARC-HD ya se estudió en el trabajo anterior, las lecciones aprendidas en este clasificador se han omitido en este punto. Por otro lado, se ha comprobado que

el método de agregación propuesto (WinWV) mejora el rendimiento de WV en el caso de SLAVE y FARC-HD. La razón por la que Chi y FURIA no mejoran el rendimiento con esta agregación es que las confianzas de la clase no predicha en ambos casos es habitualmente 0. En el caso de FURIA ésto es debido al gran ajuste de los conjuntos difusos, mientras que en CHI el hecho de que las reglas tengan más antecedentes aumenta la probabilidad de que alguno de los grados de pertenencia sea 0.

El estudio completo está publicado en:

- M. Elkano, M. Galar, J. Sanz, H. Bustince, "Fuzzy Rule-Based Classification Systems for multi-class problems using binary decomposition strategies: On the influence of n-dimensional overlap functions in the Fuzzy Reasoning Method", *Information Sciences*, vol. 332, pp. 94–114, 2016.

5.3. CHI-BD: un nuevo sistema de clasificación basado en reglas difusas para problemas de clasificación Big Data

Este trabajo representa el punto de partida de nuestro proyecto para solucionar los problemas de escalabilidad de los SCBRDs en el ámbito del Big Data. Por ello, hemos elegido el algoritmo de Chi por su simpleza e idoneidad para emplear el paradigma MapReduce. La primera adaptación de este SCBRD a MapReduce fue Chi-FRBCS-BigDataCS [LdRBH15]. Este método consiste en entrenar un clasificador Chi secuencial en cada mapper y agregar todas las bases de reglas generadas para obtener el modelo final. El gran inconveniente de esta solución es el mismo que está presente en la mayoría de metodologías que se basan en aplicar múltiples optimizaciones locales para obtener una solución global aproximada. El hecho de que cada clasificador calcule los pesos de las reglas considerando únicamente un subconjunto de ejemplos hace que el error de aproximación dependa en gran medida de la distribución de estos subconjuntos. Por consiguiente, es probable que la calidad de los pesos disminuya a medida que aumente el número de subconjuntos (mappers), limitando el grado de paralelismo del algoritmo.

Con el objetivo de solucionar esta limitación y de recuperar el modelo original del algoritmo de Chi en Big Data, proponemos un nuevo método de aprendizaje distribuido (CHI-BD) que construye la base de reglas empleando el conjunto de entrenamiento al completo, sin introducir ningún tipo de aproximación en el cálculo de los pesos. El algoritmo está compuesto por dos fases:

1. *Construcción de las reglas candidatas.* El conjunto de entrenamiento original se divide entre todos los mappers y se genera una nueva regla para cada ejemplo (sin peso). Posteriormente todos los duplicados/conflictos se agrupan en el reducer.

2. *Calculo de los pesos y resolución de conflictos.* Todos los mappers cargan en memoria las reglas generadas en la etapa anterior y calculan los grados de emparejamiento de todas ellas con los ejemplos asociados a su partición. De esta forma, cada mapper calcula una suma parcial de los grados de emparejamiento de los ejemplos de su partición para que posteriormente los reducers sumen todas las sumas parciales y calcule el peso exacto de las reglas.

Para evaluar el rendimiento de nuestra propuesta hemos llevado a cabo un estudio empírico dividido en tres bloques:

- Análisis de escalabilidad del algoritmo Chi original: hemos ejecutado la versión secuencial del método Chi para comprobar de forma empírica la capacidad de este método para solucionar problemas Big Data.
- Comparativa respecto a la solución distribuida existente (Chi-FRBCS-BigDataCS): hemos considerado 20 problemas de clasificación Big Data del repositorio UCI[3] para comparar ambos métodos en términos de precisión y tiempos de ejecución.
- Análisis de escalabilidad de nuestra propuesta: evaluamos la eficiencia de nuestro método en el conjunto de datos más grande de los considerados en este estudio (HIGGS).

Las conclusiones obtenidas en base al estudio son las siguientes:

- Los resultados arrojados por la prueba de escalabilidad del algoritmo original de Chi son contundentes: empleando solamente el 5 % del conocido dataset HIGGS, hemos decidido detener la ejecución del algoritmo después de 18 días. Nuestra propuesta distribuida se ha ejecutado en 3 minutos utilizando 32 mappers (procesadores). De esta forma, hemos concluido que la limitación de escalabilidad del algoritmo secuencial original es evidente.
- Hemos comprobado que la precisión de Chi-FRBCS-BigDataCS disminuye considerablemente cuando se aumenta el número de mappers. Así mismo, hemos validado que nuestra propuesta obtiene exactamente el mismo modelo independientemente del número de mappers.
- El promedio de mejora en precisión (medida con la media geométrica) obtenido por CHI-BD respecto a Chi-FRBCS-BigDataCS cuando se han empleado 32, 64, 128, y 256 mappers ha sido del 7 %, 9 %, 11 %, y 13 %, respectivamente.
- En cuanto a los tiempos ejecución, las diferencias entre ambos métodos depende del tamaño del conjunto de datos. Cuando se aborda un problema que no tiene suficientes ejemplos como para que el sobre-coste de MapReduce sea significativamente menor que el tiempo de cálculo, ambos métodos obtienen tiempos similares. Sin embargo, en los conjuntos de datos más grandes de UCI como KDD, SUSY, y HIGGS, la aceleración obtenida por parte de nuestro método es de 60X, 33X, y 1,5X, respectivamente.
- Los resultados de escalabilidad han mostrado que CHI-BD ofrece una aceleración lineal cuando se aumenta el número de mappers, pero tiene problemas para lidiar con el aumento en el tamaño del conjunto de datos. Esta vulnerabilidad viene dada por el gran número de reglas que genera el propio algoritmo de Chi y por el coste computacional $O(N \cdot L)$ asociado al cálculo del peso de las reglas, siendo N y L el número de reglas y ejemplos, respectivamente.

[3] `http://archive.ics.uci.edu/ml/datasets.html`

- A pesar de haber conseguido recuperar el modelo original de Chi en Big Data, las bases de reglas obtenidas son demasiado complejas como para ser interpretables.

El desarrollo completo de este nuevo método se ha publicado en el siguiente artículo:

- M. Elkano, M. Galar, J. Sanz, H. Bustince, "CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems", *Fuzzy Sets and Systems*, en prensa, 2017. DOI: 10.1016/j.fss.2017.07.003.

5.4. CHI-PG: algoritmo para la generación rápida de prototipos en problemas de clasificación Big Data

Como hemos visto en la Sección 3, la mayoría de las soluciones distribuidas de PR propuestas hasta la fecha presentan varias limitaciones de escalabilidad debidas principalmente a dos factores: el coste computacional cuadrático $O(N^2)$ heredado de las técnicas de PR existentes y el error de aproximación introducido por el uso de múltiples procesos de reducción locales.

En este trabajo nos basamos en el proceso de generación de reglas introducido en la sección anterior (CHI-BD) para construir un método de PR distribuido para Big Data (CHI-PG). El objetivo es conseguir una solución de complejidad lineal $O(N)$ que lleve a cabo un proceso de reducción global sin introducir ningún tipo de aproximación. Para ello proponemos generar prototipos a partir de las reglas extraídas por CHI-BD en un proceso formado por dos fases:

- Generar las reglas empleando la primera fase de CHI-BD (extracción de reglas candidatas sin pesos). Cada mapper genera una nueva regla para cada ejemplo de su partición y almacena los valores de los ejemplos asociados a cada regla.
- Generar los prototipos a partir de las reglas. Para cada regla, los reducers calculan la media aritmética de los ejemplos cubiertos por la regla y generan los prototipos empleando una de las siguientes estrategias:
  - Se construye un nuevo prototipo para cada una de las clases a las que pertenecen los ejemplos cubiertos por la regla
  - Se construye un prototipo solamente para la clase con mayor representación en esa regla

El estudio empírico que hemos llevado acabo para evaluar el rendimiento de nuestro método está formado por tres bloques:

- Analizamos los resultados obtenidos por diferentes configuraciones de CHI-PG midiendo la precisión en la clasificación de *k-Nearest Neighbors* (k-NN) [Mcl04], el porcentaje de reducción, y los tiempos de ejecución.
- Consideramos un método de PR distribuido que ha sido propuesto recientemente (MRPR [TPB+15]) para comparar los resultados obtenidos por este método con los de nuestra propuesta.

- Evaluamos el rendimiento de k-NN en problemas Big Data cuando se emplea el conjunto de datos original y la versión reducida construida por CHI-PG. De esta forma queremos comprobar si nuestra propuesta es una solución candidata para lidiar con los problemas de escalabilidad de k-NN en Big Data.

La comparativa con MRPR muestra que nuestro método es considerablemente más rápido y que es capaz de mantener e incluso mejorar el rendimiento de MRPR en términos de porcentaje de reducción y de precisión en la clasificación de k-NN. Por otra parte, los resultados han revelado que el uso de CHI-PG permite que k-NN se ejecute en problemas Big Data en un tiempo razonable, consiguiendo una precisión competitiva. Sin embargo, esta metodología hereda las limitaciones de CHI-BD en los conjuntos de datos de alta dimensionalidad, ya que una explosión del número de reglas podría limitar la aplicabilidad del método.

El trabajo asociado a esta propuesta es el siguiente:

- M. Elkano, M. Galar, J. Sanz, H. Bustince, "CHI-PG: A fast prototype generation algorithm for Big Data classication problems", Aceptado con revisiones menores en *Neurocomputing*, 2017.

5.5. CFM-BD: algoritmo distribuido de inducción de reglas para la construcción de Modelos Difusos Compactos en problemas de clasificación Big Data

Una vez recuperado el modelo original de Chi en Big Data, nos planteamos diseñar un algoritmo de aprendizaje distribuido más sofisticado para construir un clasificador interpretable que obtenga resultados competitivos en términos de precisión. Como hemos visto en la Sección 3, los SCBRDs diseñados para Big Data que consiguen buenos resultados en precisión generan bases de reglas demasiado complejas para ser interpretadas. Dado que la característica principal que hace destacar a los SCBRDs sobre el resto de técnicas de clasificación es la interpretabilidad, creemos que abordar esta cuestión en Big Data es de vital importancia.

Para conseguir nuestro objetivo, proponemos un nuevo método llamado CFM-BD que está basado en 3 fases:

1. *Pre-procesamiento y particionamiento difuso.* Se aplica una transformación al conjunto de entrenamiento para que todas sus variables sigan una distribución normal. Para ello se aplica la *transformada integral de probabilidad* [Ang94, Que04], la cual afirma que cualquier distribución de probabilidad puede transformarse en una distribución uniforme utilizando su función de distribución acumulada (CDF). Posteriormente se crean los conjuntos difusos sobre el conjunto de datos transformado empleando funciones de pertenencia triangulares uniformemente distribuidas a lo largo del rango de cada variable. A partir de este punto el algoritmo trabaja con el conjunto transformado. Si el usuario quisiera interpretar los conjuntos difusos en el espacio original bastaría con aplicar la *función cuantil* (inversa de la CDF) [NSB13], que es la función empleada habitualmente para generar números pseudo-aleatorios no uniformes a partir de una distribución uniforme. Esta fase permite generar

un número fijo de conjuntos difusos que se ajustan a la distribución real de los datos. El hecho de que todas las variables tengan el mismo número de etiquetas lingüísticas y que éstas se ajusten a la distribución de la variable es vital para conseguir modelos precisos e interpretables al mismo tiempo.

2. *Construcción de la base de reglas.* Se construye la base de reglas inicial en un proceso inspirado por el algoritmo Apriori [AS94] que emplea la primera fase de CHI-BD mostrada en la Sección 5.3. Primero se extraen los itemsets más frecuentes de las reglas (sin peso) generadas por CHI-BD. Posteriormente se podan los itemsets más grandes que no añaden poder de discriminación a los itemsets más pequeños. Finalmente los itemsets resultantes son convertidos a reglas difusas y se realiza un proceso de poda similar al del paso anterior, en este caso empleando la confianza o el peso de las reglas.

3. *Selección de reglas.* Hemos implementado nuestra propia versión distribuida del algoritmo evolutivo CHC [Esh91] para realizar una selección de las reglas más precisas de la base de reglas inicial.

En el estudio empírico llevado a cabo hemos considerado todos los clasificadores difusos de código abierto para Big Data existentes hasta la fecha: FMDT y FBDT [SMP17], Chi-Spark-RS [FAH17], y el propio CHI-BD. Para evaluar el rendimiento de nuestro método hemos realizado una comparativa en términos de precisión, tiempo de ejecución, y complejidad del modelo. Además, hemos analizado la escalabilidad de las tres fases que componen el algoritmo propuesto. Las conclusiones extraídas son las siguientes:

- En términos de precisión, CFM-BD mejora el rendimiento mostrado por CHI-BD y Chi-Spark-RS, si bien este último no pudo completar el proceso de aprendizaje en los conjuntos de datos más grandes (HIGGS, HEPMASS, SUSY) después de 48 horas de ejecución. En comparación con FMDT y FBDT, CFM-BD mantiene la precisión de clasificación por clases pero obtiene peores resultados en la precisión global de clasificación. La razón de esta pérdida de precisión global es que nuestro algoritmo ha sido diseñado para mantener la precisión para todas las clases, ya que desde nuestro punto de vista ésta es la medida que mejor representa el rendimiento de un clasificador en problemas multi-clase.

- En cuanto al tiempo de ejecución, el proceso evolutivo empleado por CFM-BD y Chi-Spark-RS hace que estos métodos sean considerablemente más lentos que las soluciones no evolutivas (FMDT/FBDT y CHI-BD).

- Respecto a la complejidad de los modelos, los resultados son claros: los métodos más precisos (FMDT y FBDT) obtienen cientos de miles o incluso millones de reglas, frente a las menos de 30 reglas que genera habitualmente CFM-BD. Además, en FMDT y FBDT el número de conjuntos difusos empleados para cada característica es variable (con una media de 13 por característica), mientras que nuestro método genera exactamente el mismo número de etiquetas lingüísticas para todas las variables (5). Por otro lado, a pesar de que Chi-Spark-RS y CHI-BD emplean 3 etiquetas por variable, el número de reglas generado por estos dos métodos es demasiado elevado para ser interpretable y ofrecen una menor precisión que CFM-BD.

El desarrollo completo de este trabajo se ha enviado a la revista *IEEE Transactions on Fuzzy Systems* como:

- M. Elkano, M. Galar, J. Sanz, E. Barrenechea, H. Bustince, "CFM-BD: a distributed rule induction algorithm for building Compact Fuzzy Models in Big Data classification problems", Enviado a *IEEE Transactions on Fuzzy Systems*, 2017.

6. Conclusiones

Las investigaciones realizadas en la presente tesis han tenido por objeto mejorar el rendimiento de los clasificadores difusos en dos escenarios diferentes: problemas multi-clase y Big Data. A continuación se describen las conclusiones extraídas en cada uno de ellos.

Dado el buen rendimiento mostrado por las estrategias de descomposición a la hora de mejorar la precisión de los métodos de clasificación en problemas multi-clase, hemos decidido aplicar esta metodología en uno de los clasificadores difusos más precisos e interpretables de la literatura: FARC-HD. Sin embargo, el estudio realizado sugiere que el uso del producto como operador de conjunción en las reglas de FARC-HD hace que las confianzas devueltas por el proceso de inferencia no sean adecuadas para la fase de agregación de OVO y OVA. Para resolverlo, hemos adaptado la inferencia de FARC-HD reemplazando el producto por un nuevo tipo de función de agregación: las *funciones de solapamiento n-dimensionales*. Estas funciones permiten adecuar las confianzas devueltas por FARC-HD a la fase de agregación de OVO y OVA. Además de la adaptación de la inferencia, hemos definido un nuevo método de agregación para OVO (WinWV) que ayuda a mejorar el rendimiento de esta sinergia. Estas soluciones han permitido aumentar la precisión de FARC-HD y obtener resultados competitivos respecto a otros clasificadores del estado-del-arte. En este mismo ámbito, hemos llevado a cabo un amplio estudio sobre el rendimiento de las soluciones propuestas para FARC-HD en diversos tipos de SCBRDs (CHI, SLAVE, y FURIA). De acuerdo con los resultados, la eficacia de las funciones de solapamiento n-dimensionales y del método WinWV depende del proceso de aprendizaje y de la estructura de las reglas de cada clasificador.

En el campo del Big Data, las contribuciones de esta tesis abordan dos problemáticas diferentes: el diseño de SCBRDs escalables y la reducción de prototipos (PR). Con el objetivo de establecer un punto de partida, hemos decidido desarrollar una nueva solución distribuida basada en MapReduce para el algoritmo de Chi (CHI-BD). La razón de haber elegido este SCBRD es que está basado en un procedimiento de construcción de reglas sencillo que puede ser adaptado directamente al paradigma MapReduce, lo que permite recuperar el modelo del algoritmo original de una manera distribuida.

Aprovechando la rapidez y la escalabilidad de CHI-BD, hemos planteado la posibilidad de emplear las reglas generadas por este clasificador para construir un algoritmo de PR para k-NN. La principal característica de esta nueva metodología es su complejidad computacional. La mayoría de las soluciones de PR distribuidas existentes heredan un coste computacional que tiende a $O(N^2)$, mientras que la nuestra tiene un coste lineal $O(N)$. Los resultados han mostrado que esta metodología obtiene un porcentaje de reducción y una precisión de clasificación comparable a

uno de los métodos del estado-del-arte (MRPR), en un tiempo significativamente menor. Además, ha mostrado ser una solución prometedora para poder aplicar k-NN en problemas de clasificación Big Data.

Continuando la línea de investigación sobre SCBRDs escalables, hemos propuesto un nuevo algoritmo de aprendizaje que aborda uno de los mayores inconvenientes que presentan los SCBRDs en Big Data: la interpretabilidad. A pesar de ser la característica fundamental de este tipo de técnicas, todavía no se ha conseguido obtener bases de reglas sencillas y precisas en problemas Big Data como SUSY, HIGGS, o HEPMASS. Los métodos difusos más precisos del estado-del-arte (FMDT y FBDT) generan cientos de miles de reglas formadas por un número elevado de etiquetas lingüísticas. Por ello, hemos propuesto un nuevo SCBRD capaz de ofrecer una precisión competitiva empleando modelos compactos interpretables. Los resultados revelan que las bases de reglas generadas por nuestro método están compuestas generalmente por menos de 30 reglas formadas por menos 3 antecedentes, empleando un número fijo de etiquetas lingüísticas por variable (5).

A modo de resumen, las contribuciones más relevantes de la presente tesis son las siguientes:

- El amplio estudio empírico realizado ha permitido mostrar las diferencias de comportamiento que presentan los diferentes SCBRDs cuando se emplean con estrategias de descomposición. El análisis llevado a cabo sugiere que las confianzas devueltas por algunos de los SCBRDs considerados en el estudio no son adecuadas para los métodos OVO y OVA. Como solución, hemos propuesto varias modificaciones que han permitido mejorar el rendimiento de uno de los SCBRDs más precisos e interpretables del estado-del-arte (FARC-HD).
- Todos nuestros métodos diseñados para Big Data aplican procesos de optimización y aprendizaje globales que emplean el conjunto de entrenamiento al completo. Esta propiedad es fundamental para poder extraer patrones valiosos que están ocultos cuando se consideran subconjuntos de los datos. Además, al contrario que los métodos basados en la agregación de múltiples soluciones óptimas locales, los modelos obtenidos no dependen del grado de paralelismo empleado.
- Hemos conseguido obtener un clasificador interpretable para Big Data que logra una precisión competitiva respecto a los dos clasificadores difuso más precisos del estado-del-arte (FMDT y FBDT). De acuerdo con los resultados, las bases de reglas construidas por nuestro método están compuestas generalmente por menos de 30 reglas formadas por menos de 3 antecedentes, empleando 5 etiquetas lingüísticas por variable.

7. Líneas futuras

Concluimos esta primera parte de la memoria presentando las líneas futuras de investigación que han surgido a partir de los trabajos realizados en la presente tesis.

Extender la sinergia de SCBRDs y estrategias de descomposición para soportar SCBRDs intervalo-valorados

Uno de los problemas clave de los SCBRDs es la elección de las funciones de pertenencia de los conjuntos difusos [CHV00], debido a la incertidumbre relacionada con su definición [ACW06, Men07]. Los *conjuntos difusos intervalo-valorados* (IVFSs) [Sam75] han demostrado ser una herramienta muy apropiada para modelar la incertidumbre del sistema y la ignorancia en la definición de los términos difusos [BPB+10]. En un IVFS el grado de pertenencia de cada elemento a ese conjunto es un intervalo, en lugar de un único número. La amplitud de dicho intervalo puede considerarse como la representación de la ignorancia relacionada con la asignación de un único número al grado de pertenencia [DP08]. Basándose en este marco teórico, Sanz et al. propusieron extender y adaptar el método de aprendizaje e inferencia de FARC-HD para construir un nuevo SCBRD llamado IVTURS que aprovecha las propiedades de los IVFSs [SFBH13]. Este clasificador ha mostrado ser más preciso que varios métodos del estado-del-arte, incluido el propio FARC-HD.

Además de los IVFSs, otra metodología que ha permitido mejorar la precisión de los SCBRDs son las estrategias de descomposición OVO y OVA (Secciones 5.1 y 5.2). Sin embargo, estas técnicas no están adaptadas para trabajar con intervalos y por tanto no soportan clasificadores como IVTURS. Creemos que podría ser interesante combinar el uso de IVFSs y las estrategias de descomposición para aprovechar las ventajas que ofrecen ambas aproximaciones. Para ello sería necesario adaptar los métodos de agregación de OVO y OVA para poder agregar intervalos en lugar de valores numéricos individuales.

Desarrollar una nueva versión de CHI-PG para adaptar la granularidad en tiempo real

Las dos propiedades que hacen destacar a CHI-PG sobre otras soluciones de PR para Big Data son la sencillez y la rapidez. Sin embargo, el hecho de que el número de etiquetas lingüísticas (granularidad) empleadas no dependa del problema hace que en ocasiones el método ignore ciertos ejemplos valiosos para k-NN. Este comportamiento viene derivado de la forma en que CHI-PG genera los prototipos. Este algoritmo considera que una regla difusa representa una región concreta del espacio de entrada, definida por los intervalos en los que las etiquetas lingüísticas que componen la regla tienen un grado de pertenencia mayor que 0,5. Por consiguiente, el solapamiento entre las clases representadas en dicha región suele aumentar a medida que la granularidad disminuye [CHV00, FdRBH17]. Dado que los prototipos generados van a ser utilizados por k-NN para realizar tareas de clasificación, minimizar el solapamiento entre las clases en las diferentes regiones permite obtener prototipos más representativos que ayuden a mejorar la precisión de este clasificador.

Por este motivo sería interesante diseñar una extensión de CHI-PG que adapte la granularidad de las reglas en tiempo real, basándose por ejemplo en la entropía de cada región. Si bien esta aproximación reduciría el porcentaje de reducción del método original, la mejora en precisión obtenida en k-NN ofrecería un buen equilibrio entre el tamaño del conjunto reducido y la precisión de clasificación.

Diseñar un método de particionamiento difuso escalable para Big Data

El método de particionamiento que ha mostrado el mejor rendimiento para problemas de clasificación Big Data es el empleado en los trabajos de Segatori et al. [SMP17,SBDM17]. Se trata de una solución distribuida que utiliza medidas de entropía difusa para generar el número óptimo de etiquetas lingüísticas y ajustarlas a la distribución de los datos. Sin embargo, los resultados experimentales han mostrado que el número de etiquetas lingüísticas construidas habitualmente por este método es demasiado elevado como para mantener la interpretabilidad del modelo.

Tal y como hemos descrito en la Sección 5.5, el método de particionamiento distribuido incluido en CFM-BD aplica la transformada integral de probabilidad para generar un número fijo de etiquetas lingüísticas que se ajustan a la distribución real de los datos. A pesar de que los conjuntos difusos son construidos sobre un espacio transformado, los puntos que definen la función triangular de pertenencia pueden ser recuperados en el espacio original mediante la función cuantil (inversa de la CDF), manteniendo la interpretabilidad. Esta propiedad permite que las particiones obtenidas por CFM-BD puedan ser utilizadas en cualquier clasificador difuso. Por ello, nos gustaría llevar a cabo un amplio estudio empírico que analice el comportamiento de esta metodología en los diferentes tipos de clasificadores difusos diseñados para Big Data.

Estudiar la escalabilidad de CFM-BD en problemas de alta dimensionalidad

Un serio inconveniente que presenta CFM-BD (y en general la mayoría de clasificadores difusos) es la dificultad de abordar problemas de alta dimensionalidad. En el caso concreto de CFM-BD y CHI-BD, la explosión de reglas candidatas causada por un alto número de características impediría el proceso de aprendizaje. Por ejemplo, la fase de búsqueda de itemsets presentes en el problema podría tener que procesar hasta $\sum_{len=1}^{maxLen} \binom{F \cdot L}{len}$ itemsets, siendo F y L el número de características y etiquetas lingüísticas, respectivamente, y $maxLen$ la longitud máxima de las reglas. Como vemos, para 70 características, 5 etiquetas lingüísticas, y una longitud de regla máxima de 3, el número de itemsets procesados podría ser de hasta 7.146.125, si bien generalmente el número de itemsets presentes es mucho menor que el número total.

Por este motivo, sería interesante estudiar empíricamente la escalabilidad real de los SCBRDs en problemas Big Data de alta dimensionalidad, centrándonos en mejorar el rendimiento de nuestro método (CFM-BD).

8. Introduction and conclusions (English version)

8.1. Introduction

The most primitive behaviors of human beings are determined to a great extent by reward expectations mainly driven by dopaminergic circuits [Gli11]. The human brain is continuously making predictions about the environment based on multiple sensory inputs. When new stimuli are received, the brain processes all incoming information to give a response that maximizes the reward. For instance, if we think about gripping and moving an apple towards the mouth, the sensory information mostly coming from the eyes and the mechanoreceptors located in the hand is processed to apply an adequate pressure in a certain location. In this case, the response would be the action of grabbing and moving the apple and the reward would be given by the nutrients acquired thanks to that action. However, the brain can not evaluate the actual reward before having carried out the action itself. Hence, it needs prediction mechanisms that roughly anticipates the reward associated with certain actions. This prediction is called *expectation*. This way, the success of an individual in this context will be determined by the similarity between actual and expected rewards. As time goes by, the brain adjusts this predictive model based on mistakes made in the past (experience) and modulates its responses consequently. This adaptation process is called *learning*.

In the field of Artificial Intelligence, *machine learning* [Alp04] tries to create mathematical and computational models that allow machines to build predictive systems in an automatic fashion. Similarly to reward expectations in humans, machines make predictions based on input information and evaluate the error made, which will be used to learn from experience and adapt the predictive model. In machine learning, there exist different types of predictions according to their nature, such as *classification* and *regression*. In classification tasks, the system (called *classifier*) identifies and classifies the input stimulus (data) and predicts the class it belongs to. For instance, an autonomous car must be able to identify Stop signs in the images received from the cameras [ZLZ+16]. An automatic medical diagnosis tool is able to classify the cognitive profile of Parkinson patients based on their medical history and electroencephalogram signals [MSFI+14]. In both examples, the system's prediction is a category, class, or concept. In regression tasks, the prediction made by the system are continuous. Continuing with the example of autonomous cars, once the Stop sign has been detected, the car must apply a certain force to slow down at an appropriate pace and stop smoothly. The prediction in this case is the magnitude of a force, that is, a real number.

In addition to the different types of predictions, machine learning can be *supervised*, *unsupervised*, or based on *reinforcement*. In both supervised and unsupervised learning, the system has a *training set* (representing the experience) composed of a number of *examples* (also known as instances or transactions). In the example of Parkinson's disease, an example would represent the medical history of a certain patient. The difference between these two types of learning is that supervised methods know the right prediction (output) for each training example, while such output is unknown in unsupervised tasks. This ideal output is provided by a human known as the *expert*. In this manner, supervised learning consists in fitting a predictive model that minimizes

the error in the training set, while maximizing the generalization capability of the system. This capability is essential to make accurate predictions when receiving unseen examples. In the case of unsupervised tasks, the goal is to find and extract features and properties that determine the internal structure of training examples. As in supervised methods, generalization capability is key to learning structures that are still valid for future examples. Finally, reinforcement learning replaces (or complements) the training set with stimuli received from the environment, which are used as a feedback that modulates and adjusts future responses.

In this thesis we consider only supervised methodologies for classification tasks. Although there exists a wide variety of algorithms with different learning methods and predictive models, we focus mainly on Fuzzy Rule-Based Classification Systems (FRBCSs) [INN04]. The main advantage of these systems is that they provide models consisting in a number of rules composed of human-readable linguistic labels, which allows them to explain the reasoning of the predictions made. For instance, if we train a FRBCS to distinguish different profiles of cognitive impairment in Alzheimer patients, we could obtain several rules similar to "if the patient is older and tau protein levels are very high, then the patient suffers from dementia" or "if the patient is young and tau protein levels are high, then the patient has mild cognitive impairment". Note that these contains linguistic terms such as "older", "young", "very high", or "high", concepts that are ambiguous per se. Such ambiguity allows FRBCSs not only to the reason for their decisions, but also to handle the uncertainty coming from ambiguous information. Thanks to these properties, FRBCSs have been employed in a wide range of real-world applications, including bioinformatics [HHCH06], medicine [SGJ+13], cybersecurity [TKW07], finance [SBH+14], image processing [NSY07], and traffic congestion prediction [ZOP+14].

Classification problems can be divided into two groups based on the number of classes considered: binary (two classes) and multi-class (more than two classes). In general, multi-class problems involve complex decision boundaries that are more difficult to learn than in multi-class problems, due to the higher number of classes. An effective way to deal with this situation is to decompose the original multi-class problem into easier-to-solve binary problems [GFB+11, LCG08]. Next, an independent classifier is trained in each subproblem so that each classifier specializes in distinguishing between two classes. When classifying a new example, all these classifiers make a prediction based on their experience and all the predictions are aggregated to make the final decision. This methodology has improved the performance of those classifiers that tackle multi-class problems directly [FÖ2, RK04, GFB+11, GFBH14] and has been shown to be effective for FRBCSs as well [IYN05, HB08, HH09a, FCB+10, SH11]. However, the use of decomposition strategies in FRBCSs poses a new problem: dealing with different rule structures and fuzzy reasoning methods (FRMs). The structural differences in rules are given by the wide variety of rules construction methods proposed in the literature. Such methods may differ, for example, in the type of linguistic labels generated, the conjunction/disjunction operator applied in rules composed of more than one antecedent, or the average rule length. Besides rule structures, the FRM responsible for inferring the adequate output from the rule base may notably vary from one FRBCS to another. Due to these factors, the behavior of decomposition techniques depends to a great extent on the FRBCS itself. As a result, some of the most popular aggregation methods are not able to harness the potential shown in other types of classifiers.

In addition to the added difficulty of multi-class problems, in the last few years machine learning algorithms have been struggling with a new challenge: the vast and increasing amount of information produced and consumed by the human being, also known as *Big Data*. According to the study carried out by Gantz and Reinsel in 2012 [GR12], the amount of available data will be almost doubled every two years from 2012 to 2020. Although there exist several definitions of the term "Big Data" [Art13, GH15, NMS+15], in this thesis we consider Big Data problems as those situations in which the amount of information to be processed exceeds the computing and storage capabilities of a commodity computer. Nevertheless, there is no universal threshold from which a certain amount of information should be considered as Big Data, since it depends on the type of task performed with such data. For instance, the computational complexity of traditional descriptive statistical methods such as the calculation of averages, deviations, histograms, etc. is much lower than that of machine learning methods. If we consider a 10GB dataset of medical histories, the computation of the average age of such patients might be feasible within a reasonable period of time, while the pharmacological pattern extraction might require unacceptable periods of time. Furthermore, a commodity computer equipped with an 8GB RAM memory would not be able to store all the available information in the main memory y could not even run the algorithm. In this new scenario, machine learning techniques that have often been employed are no longer feasible. Among existing solutions, *distributed computing* [GGL03, DG08] has become one of the most popular methodologies to work in Big Data environments. This solution consists in dividing the original dataset into several subsets that are distributed across a number of *nodes*. A node might be either a desktop computer or a dedicated server. Such nodes are connected to each other forming a (usually local area) network known as a *cluster*. When processing a dataset, each node is responsible for processing the subset (partition) that has been assigned to the node. Next, all the partial results are aggregated and the final result is obtained. Although this methodology resolves the issues associated with computational and storage limitations, distributed processing requires new methods that support such functionality. In the case of the FRBCSs designed for Big Data, they have not been able to maintain the performance shown until the Age of Big Data.

Another (complementary) solution to handle large-scale datasets is to apply Prototype Reduction (PR) techniques [NL11]. The goal of PR is to build a reduced version of the training set that improves the accuracy of future predictions and minimizes the number of training examples [KG15]. In other words, PR methods allows those machine learning algorithms that have not been designed for Big Data to tackle large classification problems by building a reduced version of the dataset. Moreover, several studies have shown that such reduction process helps classifiers improve their accuracy even on small datasets [GLH14]. However, the majority of the PR approaches proposed so far have serious scalability limitations that negatively affect their efficiency. Paradoxically, the computational and storage requirements that characterizes machine learning algorithms are also present in many of existing PR approaches [GLH14]. As a result, the benefit obtained from the reduction process in terms of memory and time consumption drops drastically, though accuracy might be improved anyway.

The goal of this thesis is to propose novel methodologies for improving the performance of FRBCSs in the aforementioned scenarios: multi-class and Big Data problems. To this end, we focus on the design of three types of solutions, one of them being oriented to multi-class problems and the

remaining focusing on Big Data environments. Regarding multi-class problems, we have studied and analyzed the impact of different learning and fuzzy reasoning methods of several FRBCSs on the performance of decomposition strategies. Once we have identified some of the issues associated with this synergy, we have proposed an adaptation of the FRM that improves its performance. In the case of Big Data methodologies, we have presented two new distributed learning algorithms for FRBCSs that overcome some of the limitations shown by existing methods. Additionally, we have leverage one of these algorithms to develop a new PR method with linear time complexity. In order to describe the aforementioned solutions, we have divided the dissertation into two parts:

- Part I. Devoted to the problem statement, the discussion of the experimental results, and the conclusions
- Part II. Contains the publications associated with this study.

In Part I, we present the problem statement and the techniques used (Section 2), the open problems that motivate the accomplishment of this dissertation (Section 3), and the objectives established in it (Section 4). Afterwards, we summarize the different studies carried out along this dissertation, highlighting the most important results obtained and their conclusions (Section 5). Finally, we present the conclusions of this dissertation (Section 6) and we end with the future research lines that remain to be addressed (Section 7).

In Part II, we present the set of publications associated with this dissertation:

- *Enhancing Multiclass Classification in FARC-HD Fuzzy Classifier: On the Synergy Between n-Dimensional Overlap Functions and Decomposition Strategies*
- *Fuzzy Rule-Based Classification Systems for multi-class problems using binary decomposition strategies: On the influence of n-dimensional overlap functions in the Fuzzy Reasoning Method*
- *CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems*
- *CHI-PG: A fast prototype generation algorithm for Big Data classification problems*
- *CFM-BD: a distributed rule induction algorithm for building Compact Fuzzy Models in Big Data classification problems*

8.2. Conclusions

The purpose of the research conducted in this thesis is to improve the performance of fuzzy classifiers in two different scenarios: multi-class and Big Data problems. Below we describe the conclusions extracted from each study.

Given the effectiveness of decomposition strategies to improve the accuracy of classifiers in multi-class problems, we have decided to apply this methodology with one of the most accurate and interpretable fuzzy classifiers in the literature: FARC-HD. However, the study carried out

suggests that the use of the product as conjunction operator in the rules of FARC-HD make the confidences returned by the inference process unsuitable for the aggregation stage in OVO and OVA. In order to fix it, we have adapted the inference of FARC-HD by replacing the product with a new type of aggregation function: the *n-dimensional overlap functions*. These functions allows FARC-HD to adapt its confidences to the aggregation process of OVO and OVA. In addition to this adaptation, we have defined a new aggregation method of OVO (WinWV) that enhances this synergy. These solutions have improved the accuracy of FARC-HD and provided competitive results with respect to other state-of-the-art classifiers. Following this methodology, we have carried out a broad study of the performance of the aforementioned solutions in several types of FRBCSs (Chi, SLAVE, and FURIA). According to the results, the effectiveness of n-dimensional overlap functions and WinWV depends on the learning process and the rule structure of each classifier.

In the field of Big Data, the contributions of this thesis address two different problems: the design of scalable FRBCSs and prototype reduction (PR). As a starting point, we have decided to develop a new distributed solution based on MapReduce for the Chi algorithm (CHI-BD). We have selected this FRBCS because it is based on a simple rule construction process that can be directly adapted to the MapReduce paradigm, which allows us to recover the model of the original algorithm in a distributed fashion.

Taking advantage of the speed and the scalability of CHI-BD, we have proposed using the rules generated by this classifier to develop a PR method for k-NN. The main advantage of this new methodology is its computational complexity. The majority of existing distributed PR solutions inherit a time complexity that tends to $O(N^2)$, while ours have a linear time complexity $O(N)$. The experimental results reveal that this methodology obtains comparable reduction and accuracy rates with respect to a state-of-the-art method (MRPR), in a significantly shorter period of time. Moreover, it has been shown to be a candidate solution to apply k-NN in Big Data classification problems.

Continuing the research on scalable FRBCSs, we have presented a new learning algorithm that tackles a serious issue of FRBCSs in Big Data environments: interpretability. Although it is the main feature of these methods, the rule bases obtained so far in Big Data problems such as SUSY, HIGGS, or HEPMASS are not compact enough to be interpreted. The most accurate fuzzy classifiers in the literature (FMDT and FBDT) generate hundreds of thousands of rules composed of a large number of linguistic labels. For this reason, we have proposed a new FRBCS that provides competitive accuracy and builds compact and interpretable models. The experimental results show that the rule bases generated by our method are usually composed of less than 30 rules containing less than 3 antecedents and using a fixed number of linguistic labels for all variables (5).

Next, we sum up the most relevant contributions of this thesis:

- The broad study carried out has allows us to show the differences in the performance of several types of FRBCSs when applying decomposition strategies. The experimental results suggest that the confidences returned by some FRBCSs considered in the study are inadequate for OVO and OVA. As a solution, we have proposed a number of modifications

that have enhanced one of the most accurate and interpretable state-of-the-art FRCBSs (FARC-HD).

- All our methods designed for Big Data apply global learning and optimization processes that employ the whole training set. This property is essential to extract valuable patterns that are hidden when considering subsets of data. Furthermore, contrary to those methods consisting in the aggregation of multiple locally optimal solutions, our models do not depend on the degree of parallelism used for the execution.
- We have managed to build an interpretable classifier for Big Data that achieves competitive accuracy with respect to the two most accurate fuzzy classifiers in the literature (FMDT and FBDT). According to the experimental results, the rule bases generated by our method are usually composed of less than 30 rules containing less than 3 antecedents, using 5 linguistic labels for each variable.

Bibliografía

[AA10] Aran O. y Akarun L. (2010) A multi-class classification strategy for Fisher scores: Application to signer independent sign language recognition. *Pattern Recognition* 43(5): 1776–1788.

[AAFH07] Alcalá R., Alcalá-Fdez J., y Herrera F. (2007) A proposal for the genetic lateral tuning of linguistic fuzzy systems and its interaction with rule selection. *IEEE Transactions on Fuzzy Systems* 15(4): 616–635.

[ACW06] Au W., Chan K., y Wong A. (2006) A fuzzy approach to partitioning continuous attributes for classification. *IEEE Transactions on Knowledge and Data Engineering* 18(5): 715–719.

[AFAH11] Alcalá-Fdez J., Alcalá R., y Herrera F. (2011) A Fuzzy Association Rule-Based Classification Model for High-Dimensional Problems With Genetic Rule Selection and Lateral Tuning. *IEEE Transactions on Fuzzy Systems* 19(5): 857–872.

[Alp04] Alpaydin E. (2004) *Introduction to Machine Learning*. The MIT Press.

[AMMR95] Anand R., Mehrotra K., Mohan C. K., y Ranka S. (1995) Efficient classification for multiclass problems using modular neural networks. *IEEE Transactions on Neural Networks* 6(1): 117–124.

[Ang94] Angus J. E. (1994) The Probability Integral Transform and Related Results. *SIAM Review* 36(4): 652–654.

[AP94] Aamodt A. y Plaza E. (1994) Case-based reasoning; Foundational issues, methodological variations, and system approaches. *AI Communications* 7(1): 39–59.

[Art13] Arthur L. (2013) What Is Big Data? *Forbes* .

[AS94] Agrawal R. y Srikant R. (1994) Fast Algorithms for Mining Association Rules in Large Databases. En *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, páginas 487–499. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[AS09] Anand A. y Suganthan P. N. (2009) Multiclass cancer classification by support vector machines with class-wise optimized genes and probability estimates. *Journal of Theoretical Biology* 259(3): 533–540.

[BPB+10] Bustince H., Pagola M., Barrenechea E., Fernandez J., Melo-Pinto P., Couto P., Tizhoosh H. R., y Montero J. (2010) Ignorance functions. An application to the calculation of the threshold in prostate ultrasound images. *Fuzzy Sets and Systems* 161(1): 20–36.

[CB91] Clark P. y Boswell R. (1991) Rule Induction with CN2: Some Recent Improvements. En *EWSL'91: Proc. of the European Working Session on Machine Learning*, páginas 151–163. London, UK.

[CdJH99] Cordón O., del Jesus M., y Herrera F. (1999) A proposal on reasoning methods in fuzzy rule-based classification systems. *International Journal of Approximate Reasoning* 20(1): 21–45.

[CHL05] Cano J., Herrera F., y Lozano M. (2005) Stratification for scaling up evolutionary prototype selection. *Pattern Recognition Letters* 26(7): 953–963.

[CHL07] Cano J., Herrera F., y Lozano M. (2007) Evolutionary stratified training set selection for extracting classification rules with trade off precision-interpretability. *Data & Knowledge Engineering* 60(1): 90–108.

[CHV00] Cordón O., Herrera F., y Villar P. (2000) Analysis and guidelines to obtain a good uniform fuzzy partition granularity for fuzzy rule-based systems using simulated annealing. *International Journal of Approximate Reasoning* 25(3): 187–215.

[CYP96] Chi Z., Yan H., y Pham T. (1996) Fuzzy algorithms with applications to image processing and pattern recognition. *World Scientific* .

[DG08] Dean J. y Ghemawat S. (2008) MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM* 51(1): 107–113.

[dHGGPdC12] de Haro-García A., García-Pedrajas N., y del Castillo J. R. (2012) Large scale instance selection by means of federal instance selection. *Data & Knowledge Engineering* 75: 58–77.

[DMS15] Ducange P., Marcelloni F., y Segatori A. (2015) A MapReduce-based fuzzy associative classifier for big data. En *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, páginas 1–8.

[DP08] Dubois D. y Prade H. (2008) An introduction to bipolar representations of information and preference. *International Journal of Intelligent Systems* 23(8): 866–877.

[EGSB17] Elkano M., Galar M., Sanz J., y Bustince H. (2017) CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems. *Fuzzy Sets and Systems* In Press.

[Esh91] Eshelman L. J. (1991) The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. volumen 1 of *Foundations of Genetic Algorithms*, páginas 265–283. Elsevier.

[FÖ2] Fürnkranz J. (2002) Round robin classification. *Journal of Machine Learning Research* 2: 721–747.

[FAH17] Fernandez A., Almansa E., y Herrera F. (2017) Chi-Spark-RS: An Spark-built evolutionary fuzzy rule selection algorithm in imbalanced classification for big data problems. En *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, páginas 1–6.

[FCB+10] Fernández A., Calderón M., Barrenechea E., Bustince H., y Herrera F. (2010) Solving mult-class problems with linguistic fuzzy rule based classification systems based on pairwise learning and preference relations. *Fuzzy Sets and Systems* 161(23): 3064–3080.

[FdRBH17] Fernández A., del Río S., Bawakid A., y Herrera F. (2017) Fuzzy rule based classification systems for big data with MapReduce: granularity analysis. *Advances in Data Analysis and Classification* 11(4): 711–730.

[FdRH16] Fernández A., del Río S., y Herrera F. (2016) A First Approach in Evolutionary Fuzzy Systems based on the lateral tuning of the linguistic labels for Big Data classification. En *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, páginas 1437–1444.

[FMS+17] Ferranti A., Marcelloni F., Segatori A., Antonelli M., y Ducange P. (2017) A distributed approach to multi-objective evolutionary generation of fuzzy rule-based classifiers from big data. *Information Sciences* 415-416: 319–340.

[Fri96] Friedman J. (1996) Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University.

[Für03] Fürnkranz J. (2003) Round robin ensembles. *Intelligent Data Analysis* 7(5): 385–403.

[GCH08] García S., Cano J., y Herrera F. (2008) A memetic algorithm for evolutionary prototype selection: A scaling up approach. *Pattern Recognition* 41(8): 2693–2709.

[GDCH12] García S., Derrac J., Cano J., y Herrera F. (2012) Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34(3): 417–435.

[GFB+11] Galar M., Fernández A., Barrenechea E., Bustince H., y Herrera F. (2011) An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition* 44(8): 1761–1776.

[GFB+13] Galar M., Fernández A., Barrenechea E., Bustince H., y Herrera F. (2013) Dynamic classifier selection for One-vs-One strategy: Avoiding non-competent classifiers. *Pattern Recognition* 46(12): 3412–3424.

[GFBH14] Galar M., Fernández A., Barrenechea E., y Herrera F. (2014) Empowering difficult classes with a similarity-based aggregation in multi-class classification problems. *Information Sciences* 264: 135–157.

[GGGP16] Gámez J. C., García D., González A., y Pérez R. (2016) On the use of an incremental approach to learn fuzzy classification rules for big data problems. En *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, páginas 1413–1420.

[GGL03] Ghemawat S., Gobioff H., y Leung S. (2003) The Google File System. En *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, páginas 29–43. ACM.

[GH15] Gandomi A. y Haider M. (2015) Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management* 35(2): 137–144.

[GLH14] García S., Luengo J., y Herrera F. (2014) *Data Preprocessing in Data Mining*. Springer Publishing Company, Incorporated.

[Gli11] Glimcher P. (2011) Understanding dopamine and reinforcement learning: The dopamine reward prediction error hypothesis. *Proceedings of the National Academy of Sciences of the United States of America* 108(3): 15647–15654.

[GP99] González A. y Perez R. (1999) SLAVE: a genetic learning system based on an iterative approach. *IEEE Transactions on Fuzzy Systems* 7(2): 176–191.

[GR12] Gantz J. y Reinsel D. (2012) THE DIGITAL UNIVERSE IN 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East.

[GU07] Guler I. y Ubeyli E. D. (2007) Multiclass Support Vector Machines for EEG-Signals Classification. *IEEE Transactions on Information Technology in Biomedicine* 11(2): 117–126.

[GVGdJC17] García-Vico A. M., González P., del Jesus M. J., y Carmona C. J. (2017) A first approach to handle fuzzy emerging patterns mining on big data problems: The EvAEFP-spark algorithm. En *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, páginas 1–6.

[HB08] Hüllermeier E. y Brinker K. (2008) Learning valued preference structures for solving classification problems. *Fuzzy Sets and Systems* 159(18): 2337–2352.

[HH09a] Hühn J. y Hüllermeier E. (2009) FURIA: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery* 19(3): 293–319.

[HH09b] Hühn J. C. y Hüllermeier E. (2009) FR3: A fuzzy rule learner for inducing reliable classifiers. *IEEE Transactions on Fuzzy Systems* 17(1): 138–149.

[HHCH06] Ho S., Hsieh C., Chen H., y Huang H. (2006) Interpretable gene expression classifier with an accurate and compact fuzzy rule base for microarray data analysis. *Biosystems* 85(3): 165–176.

[HMCC08] Hong J., Min J., Cho U., y Cho S. (2008) Fingerprint Classification Using One-vs-all Support Vector Machines Dynamically Ordered with Naïve Bayes Classifiers. *Pattern Recognition* 41(2): 662–671.

[HV10] Hüllermeier E. y Vanderlooy S. (2010) Combining predictions in pairwise classification: An optimal adaptive voting strategy and its relation to weighted voting. *Pattern Recognition* 43(1): 128–142.

[HZRS15] He K., Zhang X., Ren S., y Sun J. (2015) Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. En *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, páginas 1026–1034.

[INN04] Ishibuchi H., Nakashima T., y Nii M. (2004) *Classification and modeling with linguistic information granules: Advanced approaches to linguistic Data Mining*. Springer-Verlag.

[IY05] Ishibuchi H. y Yamamoto T. (2005) Rule Weight Specification in Fuzzy Rule-Based Classification Systems. *IEEE Transactions on Fuzzy Systems* 13: 428–435.

[IYN05] Ishibuchi H., Yamamoto T., y Nakashima T. (2005) Hybridization of Fuzzy GBML Approaches for Pattern Classification Problems. *IEEE Transactions on System, Man and Cybernetics B* 35(2): 359–365.

[KG15] Kuncheva L. I. y Galar M. (2015) Theoretical and Empirical Criteria for the Edited Nearest Neighbour Classifier. En *2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA, November 14-17, 2015*, páginas 817–822.

[KL06] Kavsek B. y Lavrac N. (2006) Apriori-sd: Adapting association rule learning to subgroup discovery. *Applied Artificial Intelligence* 20(7): 543–583.

[KPD90] Knerr S., Personnaz L., y Dreyfus G. (1990) Single-Layer Learning Revisited: A Stepwise Procedure for Building and Training a Neural Network. En Fogelman Soulié F. y Hérault J. (Eds.) *Neurocomputing: Algorithms, Architectures and Applications*, volumen F68 of *NATO ASI Series*, páginas 41–50. Springer-Verlag.

[KSH12] Krizhevsky A., Sutskever I., y Hinton G. E. (2012) ImageNet Classification with Deep Convolutional Neural Networks. En *Advances in Neural Information Processing Systems 25*, páginas 1097–1105.

[LCG08] Lorena A. C., Carvalho A. C., y Gama J. M. (2008) A review on the combination of binary classifiers in multiclass problems. *Artificial Intelligence Review* 30(1-4): 19–37.

[LdRBH15] López V., del Río S., Benítez M., y Herrera F. (2015) Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data. *Fuzzy Sets and Systems* 258(0): 5–38.

[LX09] Liu K. H. y Xu C. G. (2009) A genetic programming-based approach to the classification of multiclass microarray datasets. *Bioinformatics* 25(3): 331–337.

[Mam74] Mamdani E. (1974) Applications of fuzzy algorithm for control a simple dynamic plant. *Proceedings of the IEEE* 121(12): 1585–1588.

[Mcl04] Mclachlan G. J. (2004) *Discriminant Analysis and Statistical Pattern Recognition (Wiley Series in Probability and Statistics)*. Wiley-Interscience.

[Men07] Mendel J. M. (2007) Computing with words and its relationships with fuzzistics. *Information Sciences* 177(4): 988–1006.

[MSFI+14] Marin J., Soria-Frisch A., Ibáñez D., Dunne S., Grau C., Ruffini G., Rodrigues-Brazète J., Postuma R., Gagnon J.-F., Montplaisir J., y Pascual-Leone A. (2014) Advanced Machine Learning for classification of EEG traits as Parkinson's biomarker. *Frontiers in Neuroinformatics* (71).

[NL11] Nanni L. y Lumini A. (2011) Prototype reduction techniques: A comparison among different approaches. *Expert Systems with Applications* 38(9): 11820–11828.

[NMS+15] Nativi S., Mazzetti P., Santoro M., Papeschi F., Craglia M., y Ochiai O. (2015) Big Data challenges in building the Global Earth Observation System of Systems. *Environmental Modelling & Software* 68(0): 1–26.

[NSB13] Nair N. U., Sankaran P. G., y Balakrishnan N. (2013) *Quantile-Based Reliability Analysis*, Capítulo: Quantile Functions, páginas 1–28. Springer New York, New York, NY.

[NSY07] Nakashima T., Schaefer G., y Yokota Y. (2007) A weighted fuzzy classifier and its application to image processing tasks. *Fuzzy Sets and Systems* 158(3): 284–294.

[PRRRPG+17] Pulgar-Rubio F., Rivera-Rivas A., Pérez-Godoy M., González P., Carmona C., y del Jesus M. (2017) MEFASD-BD: Multi-objective evolutionary fuzzy algorithm for subgroup discovery in big data environments - A MapReduce solution. *Knowledge-Based Systems* 117(Supplement C): 70–78.

[Que04] Quesenberry C. P. (2004) *Probability Integral Transformations*. John Wiley & Sons, Inc.

[Qui86] Quinlan J. R. (1986) Induction of decision trees. *Machine Learning* 1(1): 81–106.

[RK04] Rifkin R. y Klautau A. (2004) In Defense of One-Vs-All Classification. *Journal of Machine Learning Research* 5: 101–141.

[RZGP17] Romero-Zaliz R., González A., y Pérez R. (2017) Incremental fuzzy learning algorithms in big data problems: A study on the size of learning subsets. En *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, páginas 1–6.

[Sam75] Sambuc R. (1975) *Function ö-flous, application a l'aide au diagnostic en pathologie thyroidienne*. PhD thesis, Univ. Marseille.

[SBDM17] Segatori A., Bechini A., Ducange P., y Marcelloni F. (2017) A Distributed Fuzzy Associative Classifier for Big Data. *IEEE Transactions on Cybernetics* PP(99): 1–14.

[SBH+14] Sanz J., Bernardo D., Herrera F., Bustince H., y Hagras H. (2014) A Compact Evolutionary Interval-Valued Fuzzy Rule-Based Classification System for the Modeling and Prediction of Real-World Financial Applications with Imbalanced Data. *IEEE Transactions on Fuzzy Systems* 23(4): 973–990.

[SFBH13] Sanz J., Fernández A., Bustince H., y Herrera F. (2013) IVTURS: A Linguistic Fuzzy Rule-Based Classification System Based On a New Interval-Valued Fuzzy Reasoning Method With Tuning and Rule Selection. *IEEE Transactions on Fuzzy Systems* 21(3): 399–411.

[SGJ+13] Sanz J., Galar M., Jurio A., Brugos A., Pagola M., y Bustince H. (2013) Medical diagnosis of cardiovascular diseases using an interval-valued fuzzy rule-based classification system. *Applied Soft Computing Journal* 20: 103–111.

[SH11] Senge R. y Hüllermeier E. (2011) Top-Down Induction of Fuzzy Pattern Trees. *IEEE Transactions on Fuzzy Systems* 19: 241–252.

[SMC12] Shah M., Marchand M., y Corbeil J. (2012) Feature Selection with Conjunctions of Decision Stumps and Learning from Microarray Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34(1): 174–186.

[SMP17] Segatori A., Marcelloni F., y Pedrycz W. (2017) On Distributed Fuzzy Decision Trees for Big Data. *IEEE Transactions on Fuzzy Systems* PP(99): 1–1.

[TCM04] Tao L., Chengliang Z., y Mitsunori O. (2004) A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics* 20(15): 2429–2437.

[TDGH12] Triguero I., Derrac J., García S., y Herrera F. (2012) A Taxonomy and Experimental Study on Prototype Generation for Nearest Neighbor Classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 42(1): 86–100.

[TGH11] Triguero I., García S., y Herrera F. (2011) Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification. *Pattern Recognition* 44(4): 901–916.

[TKW07] Tsang C., Kwong S., y Wang H. (2007) Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection. *Pattern Recognition* 40(9): 2373–2391.

[TPB+15] Triguero I., Peralta D., Bacardit J., García S., y Herrera F. (2015) MRPR: A MapReduce solution for prototype reduction in big data classification. *Neurocomputing* 150, Part A: 331–345.

[Vap98] Vapnik V. (1998) *Statistical Learning Theory*. New York: Wiley.

[Web83] Weber S. (1983) General concept of fuzzy connectives, negations and implications based on t-norms and t-conorms. *Fuzzy Sets and Systems* 11(2): 115–134.

[Wer90] Werbos P. J. (1990) Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE* 78(10): 1550–1560.

[WK91] Weiss S. M. y Kulikowski C. A. (1991) *Computer Systems That Learn*. Morgan Kauffman, San Mateo, CA.

[Zad65] Zadeh L. A. (1965) Fuzzy Sets. *Information and control* 8(3): 338–353.

[ZLZ+16] Zhu Z., Liang D., Zhang S., Huang X., Li B., y Hu S. (2016) Traffic-Sign Detection and Classification in the Wild. En *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[ZOP+14] Zhang X., Onieva E., Perallos A., Osaba E., y Lee V. C. (2014) Hierarchical fuzzy rule-based system optimized with genetic algorithms for short term traffic congestion prediction. *Transportation Research Part C: Emerging Technologies* 43(Part 1): 127–142.

Parte II. Publicaciones: trabajos publicados, aceptados y sometidos

1. Mejorando la clasificación multi-clase en el clasificador difuso FARC-HD: sobre la sinergia de funciones de solapamiento n-dimensionales y estrategias de descomposición – *Enhancing Multiclass Classification in FARC-HD Fuzzy Classifier: On the Synergy Between n-Dimensional Overlap Functions and Decomposition Strategies*

La publicación asociada a esta parte es:

- M. Elkano, M. Galar, J. Sanz, A. Fernández, E. Barrenechea, F. Herrera, H. Bustince "Enhancing Multiclass Classification in FARC-HD Fuzzy Classifier: On the Synergy Between n-Dimensional Overlap Functions and Decomposition Strategies", *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 5, pp. 1562–1580, 2015.
  - Estado: Publicado.
  - Índice de Impacto (JCR 2016): 7,671.
  - Áreas de Conocimiento:
    - Computer Science, Artificial Intelligence. Ranking 4/133 (Q1).
    - Engineering, Electrical & Electronic. Ranking 9/262 (Q1).

2. Sistemas de clasificación basados en reglas difusas para problemas multi-clase utilizando estrategias de descomposición binaria: sobre la influencia de las funciones de solapamiento n-dimensionales en el método de razonamiento difuso – *Fuzzy Rule-Based Classification Systems for multi-class problems using binary decomposition strategies: On the influence of n-dimensional overlap functions in the Fuzzy Reasoning Method*

La publicación asociada a esta parte es:

- M. Elkano, M. Galar, J. Sanz, H. Bustince "Fuzzy Rule-Based Classification Systems for multi-class problems using binary decomposition strategies: On the influence of n-dimensional overlap functions in the Fuzzy Reasoning Method", *Information Sciences*, vol. 332, pp. 94–114, 2016.
  - Estado: Publicado.
  - Índice de Impacto (JCR 2016): 4,832.
  - Área de Conocimiento: Computer Science, Information Systems. Ranking 7/146 (Q1).

Information Sciences 332 (2016) 94–114

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Fuzzy Rule-Based Classification Systems for multi-class problems using binary decomposition strategies: On the influence of *n*-dimensional overlap functions in the Fuzzy Reasoning Method

Mikel Elkano[a,b,*], Mikel Galar[a], Jose Sanz[a], Humberto Bustince[a,b]

[a] *Departamento de Automática y Computación, Universidad Pública de Navarra, 31006 Pamplona, Spain*
[b] *Institute of Smart Cities (ISC), Universidad Pública de Navarra, 31006 Pamplona, Spain*

ARTICLE INFO

Article history:
Received 25 November 2014
Revised 29 September 2015
Accepted 5 November 2015
Available online 10 November 2015

Keywords:
Fuzzy Rule-Based Classification Systems
Decomposition strategies
Overlap functions
Aggregations
One-vs-One
Multi-classification

ABSTRACT

Multi-class classification problems appear in a broad variety of real-world problems, e.g., medicine, genomics, bioinformatics, or computer vision. In this context, decomposition strategies are useful to increase the classification performance of classifiers. For this reason, in a previous work we proposed to improve the performance of FARC-HD (Fuzzy Association Rule-based Classification model for High-Dimensional problems) fuzzy classifier using *One-vs-One* (OVO) and *One-vs-All* (OVA) decomposition strategies. As a result of an exhaustive experimental analysis, we concluded that even though the usage of decomposition strategies was worth to be considered, further improvements could be achieved by introducing *n*-dimensional overlap functions instead of the product t-norm in the Fuzzy Reasoning Method (FRM). In this way, we can improve confidences for the subsequent processing performed in both OVO and OVA.

In this paper, we want to conduct a broader study of the influence of the usage of *n*-dimensional overlap functions to model the conjunction in several Fuzzy Rule-Based Classification Systems (FRBCSs) in order to enhance their performance in multi-class classification problems applying decomposition techniques. To do so, we adapt the FRM of four well-known FRBCSs (CHI, SLAVE, FURIA, and FARC-HD itself). We will show that the benefits of the usage of *n*-dimensional overlap functions strongly depend on both the learning algorithm and the rule structure of each classifier, which explains why FARC-HD is the most suitable one for the usage of these functions.

1. Introduction

Fuzzy Rule-Based Classification Systems (FRBCSs) [34] are one of the most popular methods in pattern recognition and machine learning. These systems feature a good performance while providing interpretable models by using linguistic labels in the antecedents of their rules [34]. FRBCSs have been successfully applied to a wide variety of domains, including bioinformatics [26], medical problems [46], or financial applications [44], among others.

* Corresponding author. Tel.: +34 630348676.
E-mail addresses: mikel.elkano@unavarra.es (M. Elkano), mikel.galar@unavarra.es (M. Galar), joseantonio.sanz@unavarra.es (J. Sanz), bustince@unavarra.es (H. Bustince).

http://dx.doi.org/10.1016/j.ins.2015.11.006
0020-0255/

Within classification tasks, two types of problems can be identified depending on the number of classes considered: binary (two classes) and multi-class (more than two classes) problems. In general, the classifier learning is more difficult for multi-class problems. This is due to the increased complexity in the definition of decision boundaries, caused by the higher overlapping among the different classes of the problem. Even so, real-world problems need to consider multiple classes in many cases: for instance, arrhythmias classification [40], fingerprints recognition [23], or microarrays analysis [6]. In this context, the application of decomposition strategies [20,39] is a straightforward manner for addressing multi-class problems, since they make any classifier capable of addressing these types of problems. Based on divide-and-conquer paradigm, the original multi-class problem is divided into easier-to-solve binary ones, which can be faced by independent binary classifiers called *base classifiers*.

Among decomposition strategies [39], *One-vs-One* (OVO) and *One-vs-All* (OVA) are the most common ones owing to their simplicity and accuracy. In the OVO scheme, the original problem is divided into as many binary sub-problems as possible pairs of classes, whereas in OVA as many sub-problems as classes in the original one are considered. When classifying a new instance, all base classifiers are queried and their outputs are combined to make the final decision (aggregation phase) [20]. These decomposition techniques usually obtain better results than addressing the problem directly, even when classifiers with inherent multi-class support are used [19,20,22,43].

Previous works have shown the effectiveness of decomposition strategies when working with FRBCSs [15,25,30,36]. Nevertheless, it should be borne in mind that, in these strategies, the final performance strongly depends on the outputs provided by each base classifier, since a new aggregation phase is introduced, which is not carried out when the problem is directly addressed. In our previous work [15], we showed that the outputs provided by FARC-HD (Fuzzy Association Rule-based Classification model for High-Dimensional problems) fuzzy classifier [2] were not suitable for decomposition schemes. This fact was due to the usage of the product to model the conjunction, since the aggregation of small values ended in outputs with low variation, quickly tending to zero. This effect was even more accentuated when the number of arguments (antecedents of fuzzy rules) increased, and as a consequence, those rules with more antecedents were penalized. However, these issues did not affect the baseline FARC-HD algorithm because output values were not used beyond the classification process. Otherwise, when using decomposition strategies, the previously mentioned facts became undesirable, since less knowledge was retained for the aggregation phase. Moreover, robust aggregations for OVO, such as weighted voting, obtained poor results with FARC-HD. On this account, the concept of n-dimensional overlap function was introduced in our previous work [15] with the aim of modeling the conjunction in the fuzzy rules of FARC-HD. In this manner, the values returned by base classifiers became more suitable for the aggregation phase, since they display a greater variation and they are independent of the number of arguments. This resulted in a significant increase in the final performance. Additionally, we proposed a new aggregation method for OVO (WinWV) with the aim of solving the problems of weighted voting caused by the unsuitable confidences provided by FARC-HD.

As a result of our previous work, the need for analyzing the behavior of n-dimensional overlap functions in different FRBCSs arises. More specifically, their behavior in the framework of multi-class problems using decomposition strategies must be analyzed. For this reason, in this paper we adapt the methodology presented in [15] to different FRBCSs. In order to obtain the broadest possible overview, we consider four different types of FRBCSs: Chi [12], SLAVE [25], FURIA [30], and FARC-HD [2] itself. We have selected these four classifiers as representative methods of FRBCSs since both their learning methods and their rule structure are clearly different. All of them have been adapted to use n-dimensional overlap functions in their Fuzzy Reasoning Method (FRM).

The main contributions of this work are the following:

- We analyze the performance of n-dimensional overlap functions in the four FRBCSs (CHI, SLAVE, FURIA, and FARC-HD) and we study whether the behavior shown in FARC-HD is extensible to other FRBCSs. In this manner, we aim to obtain a general overview of the behavior of these functions when they are applied to model the conjunction. Additionally, two decomposition strategies (OVO and OVA) are considered for each FRBCS.
- We study the impact of n-dimensional overlap functions on the rule bases generated in the four classifiers. As we will show, the usage of these functions does not only affect the performance of the model, but also its rule base. On this account, we analyze the average number of rules and antecedents per rule for each overlap function.
- We evaluate the performance of WinWV aggregation method (proposed to solve the problems of weighted voting with the confidences provided by FARC-HD) in the rest of FRBCSs. In order to do so, a comparison between WinWV aggregation strategy and the original weighted voting is performed considering the four FRBCSs.

In order to achieve well-founded conclusions, we carry out an empirical study considering twenty numerical datasets from the KEEL dataset repository [3] and we contrast the results obtained using non-parametric statistical tests, as suggested in the specialized literature [24]. In this study, we will analyze the influence of the usage of n-dimensional overlap functions when tackling directly the multi-class problem with FARC-HD, FURIA, CHI, and SLAVE baseline classifiers and when they are used as base classifiers for both OVO and OVA decomposition strategies. In all these cases, we have applied five different n-dimensional overlap functions and we have considered the usage of five aggregation strategies for OVO scheme.

The exhaustive analysis carried out shows that the benefit obtained is highly dependent on the learning process of each classifier, as well as on the structure of the rules generated after that process. The results obtained have allowed us to shed light on clarifying when it is appropriate to use n-dimensional overlap functions in the FRM of FRBCSs. That is, we explain why FARC-HD performs much better with these functions, whereas other FRBCSs present a rather different behavior.

The rest of this paper is organized as follows. Related works are reviewed in Section 2. In Section 3, we briefly describe the four FRBCSs considered in this work (FURIA, CHI, SLAVE, and FARC-HD) and we show their rule structure, learning and inference

processes. Section 4 describes OVO and OVA decomposition strategies, along with the five aggregation strategies for OVO that we use in this paper. In Section 5, we recall the concept of n-dimensional overlap function and we describe the adaptation made to model the conjunction with these functions in each FRBCS considered. The experimental framework is presented in Section 6, whereas the analysis of the results obtained is given in Section 7. Finally, Section 8 concludes this paper.

2. Related works

Fuzzy techniques are useful to achieve a trade-off between interpretability and accuracy in classification systems. In [1], authors developed a new approach to design fuzzy classifiers using k-means clustering and a memetic algorithm to find the optimal values of fuzzy rules and membership functions. Chen et al. [11] proposed a combination of a feature selection process applying modulator functions and a fuzzy rule extraction mechanism based on fuzzy clustering. In [38], authors presented a method to extract fuzzy rules from the sub-clusters produced by the output-interval clustering algorithm. Aliev et al. [4] extracted type-2 fuzzy rules applying fuzzy clustering and a Differential Evolution algorithm to optimize those rules. Finally, Sanz et al. [45] provided a framework to improve the performance of FRBCSs using interval-valued fuzzy sets.

Decomposition strategies can be considered as an ensemble method or a Multiple Classifier System (MCS), whose main objective is to enhance the classification performance using multiple classifiers. However, decomposition strategies focus on the usage of binary classifiers to address multi-class problems, whereas in ensembles and MCSs multi-class classifiers are usually considered in order to face such problems. This important difference has produced many different approaches for each type of method.

Ensemble techniques are traditionally based on creating diverse base classifiers that allow one to improve the performance as a result of the differences in their predictions, since they are complementary. Two of the most popular ensemble methods are Bagging [8] and Boosting [17], which have also been applied using fuzzy base classifiers [7,35,48]. In [7], authors proposed an extension of the classical Random Forests (a variant of bagging) making use of fuzzy decision trees. Ishibuchi and Nojima [35] combined the FRBCSs obtained in the Pareto front of a multi-objective optimization GA. In [48], authors developed a methodology to build MCSs using FURIA as base classifier, addressing all the stages from its construction (bagging-based) to the final combination process. These methods take advantage of the power of fuzzy systems to deal with soft decision boundaries, obtaining highly accurate models, but they may need thousands of rules [48]. These models are essentially focused on the final accuracy of the system, and therefore their interpretability is left aside. A clear example of this type of model is FURIA [30] (described in Section 3.4), which is one of the most extended base classifiers in this framework. FURIA generates adjusted hyper-rectangles for each rule instead of using the same linguistic labels for the entire rule base, and hence it cannot be considered as interpretable as a classical FRBCS [34]. For this reason, in this paper we will only consider decomposition-based ensembles, which may partially maintain the interpretability of the baseline models.

Decomposition strategies have become a commonly used approach to improve the performance of FRBCSs in multi-class classification problems [16,30,32,47]. These strategies have been successfully applied using different base classifiers, such as Fuzzy Ripper [31], FH-GBML [36] or SLAVE [25] (described in Section 3.3). Moreover, Non-Dominance criterion (ND) [16] and Learning Valued Preference for Classification (LVPC) [30,32] aggregation strategies (described in Section 4) have been specifically proposed for these fuzzy classifiers. In both of them, preference relations are considered to model the aggregation phase, where the best alternative should be predicted. In order to do so, Hüllermeier and Brinker [32] modeled the conflict and ignorance from the outputs of the Fuzzy Ripper algorithm [31]. From a different perspective, Fernandez et al. [16] proposed the usage of ND criterion in FH-GBML and SLAVE classifiers, obtaining good results. Finally, the Top-Down induction of Fuzzy Pattern Trees (PTTD) was presented in [47], where an OVA approach was applied.

In the framework of decomposition techniques, in [15] we proposed n-dimensional overlap functions to provide more suitable confidences when combining FARC-HD fuzzy classifier and decomposition strategies, which resulted in an enhancement of the final performance of FARC-HD. Based on this work, our aim is to extend this methodology to different FRBCSs and to study the behavior of these functions when they are applied in different FRMs.

3. Fuzzy Rule-Based Classification Systems

In this section we first introduce the preliminary concepts related to FRBCSs (Section 3.1). Next, a description of all the classifiers considered in this work is shown, along with their rule structure, learning algorithms and inference methods (Sections 3.2–3.5).

3.1. Preliminary concepts

In the literature, there are multiple techniques used to solve classification problems. Among them, FRBCSs are one of the most popular approaches, since they provide an interpretable model by means of the use of linguistic labels in their rules [34].

The two main components of FRBCSs are as follows.

1. *Knowledge base*: It is composed of both the rule base (RB) and the database, where the rules and the membership functions used to model the linguistic labels are stored, respectively.
2. *Fuzzy Reasoning Method (FRM)*: This is the mechanism used to classify examples using the information stored in the knowledge base.

Table 1
Notation defined for all FRBCSs considered in this paper.

Term	Description
n	Number of variables
$\mathcal{D}_T$	Training set
P	Number of examples in the training set
x_p	pth training example
$\mathbb{C}$	Set of classes
m	Number of classes
y_p	Class of the pth training example
R_j	jth rule
n_j	Number of antecedents of the jth rule
C_j	Cass of the jth rule
$\mathbb{L}_i$	Set of linguistic labels for the ith variable
l	Number of linguistic labels in $\mathbb{L}_i$

In order to generate the knowledge base, a fuzzy rule learning algorithm is applied using a training set $\mathcal{D}_T$ composed of P labeled examples $x_p = (x_{p1}, \ldots, x_{pn})$, $p = \{1, \ldots, P\}$, where x_{pi} is the value of the ith attribute ($i = \{1, 2, \ldots, n\}$) of the pth training example. Each example belongs to a class $y_p \in \mathbb{C} = \{C_1, C_2, ..., C_m\}$, where m is the number of classes of the problem.

Since we consider multiple FRBCSs, in Table 1 we introduce the common notation to make them easier to understand.

3.2. CHI algorithm

CHI algorithm [12] generates the rule base establishing an association between variables (antecedents) and classes (consequents). The rule structure used by this algorithm is as follows:

$$\text{Rule } R_j: \text{ If } x_1 \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_n \text{ is } A_{jn} \text{ then Class} = C_j \text{ with } RW_j \tag{1}$$

where R_j is the label of the jth rule, $x = (x_1, \ldots, x_n)$ is a n-dimensional pattern vector that represents the example, $A_{ji} \in \mathbb{L}_i$ is a linguistic label modeled by a triangular membership function (being $\mathbb{L}_i = \{L_{i1}, \ldots, L_{il}\}$ the set of linguistic labels for the ith antecedent, where l is the number of linguistic labels in this set), C_j is the class label and RW_j is the rule weight computed using the most common specification, i.e., the fuzzy confidence value or certainty factor defined in [36]:

$$RW_j = CF_j = \frac{\sum_{x_p \in Class C_j} \mu_{A_j}(x_p)}{\sum_{p=1}^{P} \mu_{A_j}(x_p)} \tag{2}$$

being $\mu_{A_j}(x_p)$ the matching degree of the example x_p with the antecedent part of the fuzzy rule R_j computed as follows:

$$\mu_{A_j}(x_p) = T\left(\mu_{A_{j1}}(x_{p1}), \ldots, \mu_{A_{jn}}(x_{pn})\right) \tag{3}$$

where $\mu_{A_{ji}}(x_{pi})$ is the membership degree of the value x_{pi} to the fuzzy set A_{ji} of the rule R_j and T is a t-norm.

In order to construct the rule base, CHI applies the following learning process:

1. *Definition of the linguistic partitions.* Fuzzy partitions are constructed with the same triangular shape and equally distributed on the range of values.
2. *Generation of a fuzzy rule for each example*. A fuzzy rule is generated for each example x_p as follows.
   (a) The membership degree of each value x_{pi} to all the different fuzzy sets of the ith variable is computed.
   (b) For each variable, the linguistic label with the greatest membership degree is selected.
   (c) A rule is generated for the example where the antecedent part is determined by the selected fuzzy region, that is, the intersection of the selected linguistic labels, and the consequent is the class label of the example (y_p). Notice that in this algorithm no feature selection is performed in the learning process, and hence all rules have exactly the same number of antecedents as variables in the problem (n).
   (d) The rule weight is computed using the certainty factor given in Eq. (2).

Note that after the learning process we can obtain duplicated rules with the same antecedent part and different consequent part. In that case, only the one with the highest rule weight is kept.

In order to classify a new example x_p, in this paper we consider the usage of the *additive combination* [14] FRM, which is composed of the following steps.

1. *Matching degree*. The strength of activation of the antecedent part for all rules in the rule base with the example x_p is computed (Eq. (3)).
2. *Association degree*. The association degree of the example x_p with each rule in the rule base is computed.

$$b_j(x_p) = \mu_{A_j}(x_p) \cdot RW_j \tag{4}$$

3. *Confidence degree*. The confidence degree for each class is computed. To obtain the confidence degree of a class, the association degrees of the rules of that class, i.e., those whose consequent is the class we are considering, are summed.

$$conf_c(x_p) = \sum_{R_j \in RB;\ C_j = c} b_j(x_p), \qquad c = 1, 2, \ldots, m \tag{5}$$

4. *Classification*. The class that obtains the highest confidence degree is predicted.

$$Class = arg \max_{c=1,\ldots,m} (conf_c(x_p)) \tag{6}$$

3.3. SLAVE

SLAVE (Structural Learning Algorithm in a Vague Environment) [25] is an inductive learning algorithm that makes use of an iterative approach to learn fuzzy rules. In addition, it takes advantage of a Genetic Algorithm (GA) to reduce the number of rules, keeping only the most relevant ones for each class. The rule structure in SLAVE is as follows:

$$\text{Rule } R_j:\ \text{If } x_1 \text{ is } \mathcal{A}_{j1} \text{ and } \ldots \text{ and } x_{n_j} \text{ is } \mathcal{A}_{jn_j} \text{ then Class} = C_j \text{ with } RW_j \tag{7}$$

where $\mathcal{A}_{ji} \subseteq \mathbb{L}_i$ is a subset of linguistic labels modeled by triangular membership functions and n_j is the number of antecedents of the rule. In this case, the rule weight is computed as:

$$RW_j = \frac{n^+(R_j)}{n(R_j)} \tag{8}$$

being $n^+(R_j)$ the number of positive examples for the rule R_j and $n(R_j)$ the number of covered examples by the rule R_j (the definition of these concepts is described in detail in [25]). A short example is shown below in order to clarify these types of rules.

Example 1. A rule such as

$$R_j:\ \text{If } x_1 \text{ is } \{L_{11}, L_{13}, L_{14}\} \text{ and } \ldots \text{ and } x_{n_j} \text{ is } \left\{L_{n_j 2}, L_{n_j 4}\right\} \text{ then Class} = C_j \text{ with } RW_j$$

is equivalent to

$$R_j: \text{If } (x_1 \text{ is } L_{11} \text{ or } x_1 \text{ is } L_{13} \text{ or } x_1 \text{ is } L_{14}) \text{ and } \ldots \text{ and } \left(x_{n_j} \text{ is } L_{n_j 2} \text{ or } x_{n_j} \text{ is } L_{n_j 4}\right)$$
$$\text{then Class} = C_j \text{ with } RW_j$$

As it can be observed, there are two main differences between CHI and SLAVE regarding the rule structure. On the one side, in SLAVE the number of antecedents may vary depending on the rule (an embedded feature selection process is carried out), whereas in CHI the number of antecedents in all rules is the same (all variables are used). On the other hand, in the case of SLAVE, a single antecedent can be composed of multiple linguistic labels, while in CHI each antecedent is a single linguistic label. In order to compute the disjunction (OR operator) of linguistic labels, the membership degrees of the input value to all of them are computed. Then, the maximum of these membership degrees is taken.

As in the case of CHI, the learning algorithm of SLAVE tries to obtain a rule base that represents the relationship between antecedents and the class, keeping only those antecedents that are necessary to properly represent the class for each rule. In order to do so, SLAVE applies an iterative method for each class in $\mathbb{C}$ that works as follows.

1. Given a training set $\mathcal{D}_T$ and a class C, the algorithm selects the best rule that represents the examples belonging to C. A rule is considered to be the best if it:
   - Covers the maximum number of examples of the class C.
   - Covers the minimum number of examples of the rest of classes.

   In order to find the best rule, SLAVE applies a Genetic Algorithm (GA) to simultaneously optimize both previous criteria.
2. The examples covered by the selected rule are removed from $\mathcal{D}_T$.
3. The process is repeated until no useful rules can be extracted for the class C. This situation happens when the optimization criteria cannot be fulfilled.
4. Once all rules for a class have been extracted, the same process is repeated with the rest of classes.

In order to classify a new example x_p, the inference works as follows:

1. *Adaptation degree*. The adaptation degree between the example and the antecedent part of each rule ($U_j(x_p, A_j)$) is computed. To do so, the measures of possibility of all A_{ji} are aggregated by a t-norm (in this case the product).

$$U_j(x_p, A_j) = T\left(Poss(A_{j1}|x_{p1}), Poss(A_{j2}|x_{p2}), \ldots, Poss(A_{jn}|x_{pn_j})\right) \tag{9}$$

   The possibility measure of a given antecedent ($Poss(A_{ji}|x_{pi})$) is defined as the proportion of the maximum membership degree of the considered linguistic labels for that antecedent with respect to the maximum membership degree of all linguistic labels. The complete definition of this measure is presented in [25].
2. *Association degree*. The association degree of the example x_p with each rule in the rule base is computed.

$$b_j(x_p) = U_j(x_p, A_j) \cdot RW_j \tag{10}$$

3. *Classification*. The class of the rule with the highest association degree ($b_j(x_p)$) is predicted. If there are two or more rules with the same association degree, SLAVE applies the following criteria:
   (a) The rule with the highest rule weight is the winner.
   (b) If the rule weights are the same, the rule that covered the least number of examples is the winner (in favor of specific rules).
   (c) In case of a tie, the first learned rule is the winner.

3.4. FURIA

FURIA (Fuzzy Unordered Rule Induction Algorithm) [30] modifies and extends RIPPER rule induction algorithm [13], learning fuzzy rules instead of conventional rules and unordered rule sets instead of rule lists. The rule structure in FURIA is as follows.

$$\text{Rule } R_j: \text{ If } x_1 \text{ is } A^I_{j1} \text{ and } \ldots \text{ and } x_{n_j} \text{ is } A^I_{jn_j} \text{ then Class} = C_j \text{ with } RW_j \tag{11}$$

where A^I_{ji} is a trapezoidal membership function corresponding to the variable i defined as $A^I_{ji} = (\phi^{s,L}_{ji}, \phi^{c,L}_{ji}, \phi^{c,U}_{ji}, \phi^{s,U}_{ji})$, being $\phi^{c,L}_{ji}$ and $\phi^{c,U}_{ji}$ the lower and upper bounds of the core and $\phi^{s,L}_{ji}$ and $\phi^{s,U}_{ji}$ the lower and upper bounds of the support, respectively. With the aim of obtaining more flexible decision boundaries, FURIA applies a certainty factor to each rule (similar to the rule weight of SLAVE and CHI), which is computed as:

$$RW_j = \frac{2\frac{\left|\mathcal{D}_T^{(C_j)}\right|}{|\mathcal{D}_T|} + \sum_{x_p \in \mathcal{D}_T^{(C_j)}} \mu_{A^I_j}(x_p)}{2 + \sum_{x_p \in \mathcal{D}_T} \mu_{A^I_j}(x_p)} \tag{12}$$

where $\mathcal{D}_T$ represents the training set, $\mathcal{D}_T^{(C_j)}$ are the examples of the class of the rule (C_j), and $\mu_{A^I_j}(x_p)$ is the *coverage degree* (equivalent to the matching degree of Eq. (3)) of the rule R_j for the example x_p computed as:

$$\mu_{A^I_j}(x_p) = T\left(\mu_{A^I_{j1}}(x_{p1}), \mu_{A^I_{j2}}(x_{p2}), \ldots, \mu_{A^I_{jn_j}}(x_{pn_j})\right) \tag{13}$$

being $\mu_{A^I_{ji}}(x_{pi})$ the membership degree of the ith element and T a t-norm (in this case the product).

Looking at the rules used in FURIA, it can be observed that antecedents are not represented by triangular membership functions as in CHI and SLAVE. Instead, FURIA uses fuzzy sets with trapezoidal membership functions. We must stress that each membership function is specific to each antecedent, and thus it can be different for each fuzzy rule.

In order to generate the rule base, FURIA applies a learning algorithm composed of the following stages:

1. *Learn a rule set for each class using RIPPER algorithm*. This stage is divided into the building and optimization phases.
2. *Fuzzification of rules generated by RIPPER*. In this stage the structure of the rules is maintained, but the interval representing each antecedent is replaced by a trapezoidal membership function (Eq. (11)). To do so, the original interval of an antecedent is considered as the core ($\phi^{c,L}_{ji}, \phi^{c,U}_{ji}$) of the trapezoidal membership function, and then the optimal support bounds are adjusted. In order to solve this optimization problem, FURIA applies a greedy algorithm (in each rule) where a single antecedent i is fuzzified in each iteration, measuring the quality of that fuzzification in terms of *rule purity*. For this computation, only the relevant training data for rule j and antecedent i (D^{ji}_T) are considered:

$$\mathcal{D}^{ji}_T = \left\{x \in \mathcal{D}_T \mid \mu_{A^I_{jk}}(x_k) > 0 \text{ for all } k = 1, \ldots, n_j \text{ and } k \neq i\right\}$$

Once the relevant data have been selected, this set is further divided into two subsets:
- Positive instances (those belonging to the class of the rule), $\mathcal{D}^{ji}_{T+}$
- Negative instances (rest of instances), $\mathcal{D}^{ji}_{T-}$

Then, the rule purity is computed as follows:

$$pur_{ji} = \frac{p_{ji}}{p_{ji} + n_{ji}} \tag{14}$$

where

$$p_{ji} = \sum_{x \in \mathcal{D}^{ji}_{T+}} \mu_{A^I_{j1}}(x_i)$$

$$n_{ji} = \sum_{x \in \mathcal{D}^{ji}_{T-}} \mu_{A^I_{j1}}(x_i)$$

Note that after the fuzzification stage, each antecedent of each rule has its own trapezoidal membership function, and thus linguistic labels are not shared by all rules as in the rest of classifiers considered in this paper. Hence, FURIA makes use of fuzzy theory to improve the accuracy of the system, leaving its interpretability aside.

When classifying a new example x_p, FURIA applies the same FRM as CHI (Eqs. (3)–(6)), but using trapezoidal membership functions ($\mu_{A^l_{ji}}(x_{pi})$) instead of triangular ones. In addition, if the example is not covered by any rule, a rule generalization process (*stretching*) is carried out replacing all rules by their minimal generalizations, which are obtained removing all the antecedents that are not satisfied by the query. In case of a tie, the class with highest frequency is predicted.

3.5. FARC-HD

FARC-HD (Fuzzy Association Rule-based Classification model for High-Dimensional problems) [2] is a fuzzy association rule-based classifier. Apriori algorithm is used to learn fuzzy rules before applying a subgroup discovery technique and an Evolutionary Algorithm is used to reduce the computational cost and improve the accuracy and interpretability of the model.

This method uses the following rule structure:

$$\text{Rule } R_j: \text{ If } x_1 \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_{n_j} \text{ is } A_{jn_j} \text{ then Class} = C_j \text{ with } RW_j \tag{15}$$

where the rule weight is computed applying the certainty factor (Eq. (2)). As we can observe, the rule structure is the same as that of CHI (Eq. (1)). However, notice that in FARC-HD the number of antecedents may vary depending on the rules due to the way the latter are learned.

The learning algorithm of FARC-HD is composed of the three following stages:

1. *Fuzzy association rule extraction for classification*: In order to generate the rule base, a search tree is constructed for each class. To this end, frequent itemsets (sets of linguistic labels) are computed considering the support and confidence. Once the frequent itemsets are obtained, the fuzzy rules are extracted. The number of linguistic terms in the antecedents is limited by the maximum depth of the tree.
2. *Candidate rule pre-screening*: The most interesting fuzzy rules are pre-selected from the rule base obtained in the previous stage. To do so, a pattern weighting scheme is applied, where the weights of the examples are based on the coverage of the fuzzy rules.
3. *Genetic rule selection and lateral tuning*: An evolutionary algorithm is used both to tune the lateral position of the membership functions and to select the most accurate rules from the rule base generated in the previous steps.

In order to classify a new example, FARC-HD also applies the same FRM as CHI (Eqs. (3)–(6)).

4. Decomposition strategies

Decomposition strategies [39] divide the original multi-class problem into simpler binary problems that are faced by independent binary classifiers, which are called base classifiers. These strategies are not only useful when working with classifiers that are only capable of discriminating between two classes, but also with those having an inherent multi-class support. Even in the latter case, the results are usually enhanced when decomposition strategies are applied [19,20,22,43]. In this paper, we consider two of the most popular decomposition strategies in the literature: *One-vs-One* (OVO) and *One-vs-All* (OVA) [20] strategies.

4.1. One-vs-One (OVO)

OVO strategy divides a m class problem into $m(m-1)/2$ binary sub-problems (all the possible combinations between pairs of classes). Each binary problem is faced by an independent base classifier which distinguishes a pair of classes $\{C_i, C_j\}$. When classifying a new example, all base classifiers are queried and their outputs are collected. For each classifier, a pair of confidence degrees $r_{ij}, r_{ji} \in [0, 1]$ in favor of classes C_i and C_j, respectively, are obtained. The outputs obtained from all base classifiers are stored in a *score-matrix* R:

$$R = \begin{pmatrix} - & r_{12} & \cdots & r_{1m} \\ r_{21} & - & \cdots & r_{2m} \\ \vdots & & & \vdots \\ r_{m1} & r_{m2} & \cdots & - \end{pmatrix} \tag{16}$$

Since each binary sub-problem is addressed by an independent base classifier, the score-matrix needs to be normalized in order to have all confidence degrees within the same range of values. This normalization is important when using classifiers that do not return confidences in [0,1], which could be interpreted as probabilities (which is the case of the FRBCSs tested in this paper). The normalization of the score-matrix ($\hat{R}$) is performed as follows.

$$\hat{r}_{ij} = \begin{cases} \dfrac{r_{ij}}{r_{ij} + r_{ji}} & \text{if } r_{ij} \neq 0 \text{ or } r_{ji} \neq 0 \\ 0.5 & \text{if } r_{ij} = r_{ji} = 0 \end{cases} \tag{17}$$

Finally, the outputs of base classifiers stored in the score-matrix are aggregated and the class is predicted. This aggregation phase is a key factor for the classification success [20]. Next, we briefly describe the five well-known OVO aggregation methods that we consider in this paper.

- *Voting strategy* (VOTE) [18]. Each base classifier gives a vote for its predicted class. The class having the largest number of votes is given as output:

$$Class = arg \max_{i=1,\dots,m} \sum_{1 \le j \ne i \le m} s_{ij} \tag{18}$$

where s_{ij} is 1 if $\hat{r}_{ij} > \hat{r}_{ji}$ and 0 otherwise.
- *Weighted Voting* (WV) [33]. Each base classifier votes for both classes based on the confidence degree provided for each one. The class obtaining the highest value is given as output:

$$Class = arg \max_{i=1,\dots,m} \sum_{1 \le j \ne i \le m} \hat{r}_{ij} \tag{19}$$

- WinWV [15]. This aggregation method was proposed in our previous work [15] in order to solve the problems of WV with the confidences provided by FARC-HD. To do so, this method only considers the confidence of the predicted class, whereas that of the non-predicted class is not taken into account. Therefore, WinWV aggregation strategy works as follows:

$$Class = arg \max_{i=1,\dots,m} \sum_{1 \le j \ne i \le m} s_{ij} \tag{20}$$

where s_{ij} is $\hat{r}_{ij}$ if $\hat{r}_{ij} > \hat{r}_{ji}$ and 0 otherwise.
- *Non-Dominance Criteria* (ND) [16]. The score-matrix is considered as a fuzzy preference relation. Then the non-dominance degree is computed, being the winning class the one with the highest value:

$$Class = arg \max_{i=1,\dots,m} \left\{ 1 - \max_{j=1,\dots,m} r'_{ji} \right\} \tag{21}$$

where R' is the strict score-matrix (after normalization).
- *Learning valued preference for classification* (LVPC) {CITEHuhn2009Blhyp2,Hullermeier2008. LVPC strategy considers the score-matrix as a fuzzy preference relation, as ND does. Based on fuzzy preference modeling, the original relation is decomposed into three new relations with different meanings: strict preference, conflict, and ignorance. Finally a decision rule based on a voting strategy is proposed to obtain the output class:

$$Class = arg \max_{i=1,\dots,m} \sum_{1 \le j \ne i \le m} P_{ij} + \frac{1}{2} C_{ij} + \frac{N_i}{N_i + N_j} I_{ij} \tag{22}$$

being N_i the number of training examples belonging to class i, C_{ij} the degree of conflict (the degree to which both classes are supported), I_{ij} the degree of ignorance (the degree to which none of the classes are supported), and P_{ij} and P_{ji} the strict preference for i and j, respectively. These variables are computed as follows:

$$C_{ij} = \min\left\{\hat{r}_{ij}, \hat{r}_{ji}\right\}, \qquad P_{ij} = \hat{r}_{ij} - C_{ij}, \qquad P_{ji} = \hat{r}_{ji} - C_{ij}, \qquad I_{ij} = 1 - \max\left\{\hat{r}_{ij}, \hat{r}_{ji}\right\}$$

It should be mentioned that, from the division of the multi-class problem in OVO, an inherent problematic issue arises: the non-competent classifiers [21]. This is due to the fact that each base classifier learns the model only using the examples belonging to the two classes that it discriminates, and thus the examples belonging to the rest of classes are ignored. Consequently, the remainder classes are unknown for this classifier and its outputs will be irrelevant to classify examples of those classes even though they are aggregated, since the non-competence cannot be established a priori. Even if this circumstance should be taken into account when applying OVO strategy, this problematic lies outside the scope of this paper and shall be considered in future works.

4.2. One-vs-All (OVA)

OVA decomposition divides a m class problem into m binary sub-problems, which are faced by independent binary classifiers. Each base classifier distinguishes one of the classes from the remaining ones, learning the model using all examples of the training set. To this end, the examples of the class to be distinguished are considered as positives, whereas the rest are labeled as negatives. When classifying a new example, all base classifiers are queried and a confidence degree $\hat{r}_i \in [0, 1]$ in favor of the class C_i is returned by each classifier. The outputs of all base classifiers are stored in the *score-vector* R:

$$R = (r_1, \dots, r_i, \dots, r_m) \tag{23}$$

However, as in OVO, the range of the values returned by each classifier depends on each sub-problem. These differences among the ranges can lead us to misclassify an example, since the comparison among the confidences may not be fair. Therefore, the score-vector R needs to be normalized in such a way that all classifiers return values in the same range. To this aim, we normalize the score-vector with respect to the confidences obtained by each classifier for the negative class (stored in another score-vector $\overline{R}$). Once both vectors are obtained, the normalization of the score-vector ($\hat{R}$) is performed as follows.

$$\hat{r}_i = \frac{r_i}{r_i + \overline{r}_i} \tag{24}$$

Finally, in OVA the values of the score-vector are usually aggregated using the maximum, and thus the class with the highest confidence will be predicted. Another aggregation method for OVA is the so-called *dynamically ordered OVA* [29]. Nevertheless, in this work we only focus on the maximum because usually no statistical differences are found and the maximum is simpler.

5. Modeling the conjunction in FRBCSs with *n*-dimensional overlap functions: extending the FRMs

In our previous work [15], we showed that the confidences returned by FARC-HD are unsuitable for their subsequent processing in decomposition strategies. This was caused by the usage of the product in the FRM of FARC-HD. In order to solve this problem, we proposed to replace the product t-norm by n-dimensional overlap functions to model the conjunction in the FRM of FARC-HD. In this paper, we extend this methodology to four different FRBCSs by adapting their FRMs. In this manner, we aim to obtain a broader view of how n-dimensional overlap functions behave when they are used to model the conjunction in different FRBCSs.

In the rest of this section, we first recall the concept of n-dimensional overlap function introduced in our previous work [15] and we show the five different functions considered in this paper (Section 5.1). Next, we describe how these functions are included in the different FRBCSs in order to model the conjunction in their fuzzy rules (Section 5.2).

5.1. n-dimensional overlap functions

The original concept of overlap function [9] was introduced in image processing with the purpose of classifying those pixels whose belonging to the object or to the background was not clear. Examples of the application of these functions to image processing problems can be found in [37,42]. Furthermore, these functions were also applied to model the indifference in preference relations [10]. Due to the fact that overlap functions allow one to recover many of the characteristics of t-norms without imposing the associativity property, their application range has turned out to be much broader. Taking advantage of these properties, an extension of overlap functions was proposed in our previous work [15] in order to adapt the inference process of FARC-HD to decomposition strategies by modeling the conjunction with these functions. With this aim, we extended the original concept of two dimensional overlap function to any finite dimension n (recovering the original definition when $n = 2$).

Let us recall the definition of the original two dimensional case:

Definition 1 ([9]). A function O: $[0, 1] \times [0, 1] \to [0, 1]$ is an overlap function if satisfies the following conditions :

1. $O(x, y) = O(y, x)$ *for all* $x, y \in [0, 1]$.
2. $O(x, y) = 0$ *if and only if* $x \cdot y = 0$.
3. $O(x, y) = 1$ *if and only if* $x \cdot y = 1$.
4. O is increasing.
5. O is continuous.

Based on the previous definition, the following extension was proposed:

Definition 2 ([15]). A n-dimensional function O: $[0, 1]^n \to [0, 1]$ with $n \geq 2$ is a n-dimensional overlap function if the following properties hold:

1. O is symmetric.
2. $O(x_1, \ldots, x_n) = 0$ if and only if $\prod_{i=1}^{n} x_i = 0$.
3. $O(x_1, \ldots, x_n) = 1$ if and only if $\prod_{i=1}^{n} x_i = 1$.
4. O is increasing.
5. O is continuous.

Furthermore, a construction method for n-dimensional overlap functions using rational expressions was presented:

Theorem 1 ([15]). *The mapping* O^n: $[0, 1]^n \to [0, 1]$ *is a n-dimensional overlap function if and only if there exist* f, g: $[0, 1]^n \to [0, 1]$ *with*

$$O^n(x_1, \ldots, x_n) = \frac{f(x_1, \ldots, x_n)}{f(x_1, \ldots, x_n) + g(x_1, \ldots, x_n)}$$

where

1. *f and g are symmetric.*
2. *f is non-decreasing and g is non-increasing.*
3. $f(x_1, \ldots, x_n) = 0$ *if and only if* $\prod_{i=1}^{n} x_i = 0$.
4. $g(x_1, \ldots, x_n) = 0$ *if and only if* $\prod_{i=1}^{n} x_i = 1$.
5. *f and g are continuous.*

In this paper we have considered five different n-dimensional overlap functions:

- *Product (PROD)*: The returned value is the product of the input values. The original behavior of all FRBCSs considered in this paper are recovered.

$$O(x_1, \ldots, x_n) = \prod_{i=1}^{n} x_i \tag{25}$$

- *Minimum (MIN)*: Returns the minimum of the input values. This is a t-norm as well, but unlike the product, the returned value does not decrease when the number of arguments increases. The minimum is commonly used in FRBCSs.

$$O(x_1 \ldots, x_n) = \min(x_1, \ldots, x_n) \tag{26}$$

- *Harmonic mean (HM)*: The returned value is the harmonic mean of the input values if all of them are different from zero and 0 otherwise.

$$O(x_1, x_2, \ldots, x_n) = \begin{cases} \dfrac{n}{\frac{1}{x_1} + \ldots + \frac{1}{x_n}} & \text{if } x_i \neq 0, \quad \text{for all } i = 1, \ldots, n \\ 0 & \text{otherwise.} \end{cases} \tag{27}$$

- *Geometric mean (GM)*: Returns the geometric mean of the input values.

$$O(x_1, x_2, \ldots, x_n) = \sqrt[n]{\prod_{i=1}^{n} x_i} \tag{28}$$

- *Sine (SIN)*: This overlap function returns higher values than means. It is interesting to study the behavior of these types of functions for modeling the conjunction.

$$O(x_1, \ldots, x_n) = \sin\left(\frac{\pi}{2}\left(\prod_{i=1}^{n} x_i\right)^{\alpha}\right) \tag{29}$$

where $\alpha \leq \frac{1}{2n}$. In the experiments carried out in Section 7, we take $\alpha = \frac{1}{2n}$.

According to the values returned, we can establish an order among overlap functions. An overlap function is considered greater than other one if, for every possible input data, the values returned by the first function are higher than those returned by the second one. Among the considered overlap functions, the smallest one is the product t-norm, which returns values with a lower variation than the remaining functions when aggregating small values and whose output decreases as the number of arguments increases. Then, we have the minimum, a t-norm whose behavior is not affected by the number of arguments. Next, the harmonic and geometric means are considered (in this order) as representatives of means that return higher values than t-norms [5]. Finally, the largest function is the SIN, which returns higher values than means. The different behaviors among the considered overlap functions give us a general overview in the experiments carried out in Section 7.

In [15], we showed that those overlap functions satisfying the idempotency property provide better results, that is,

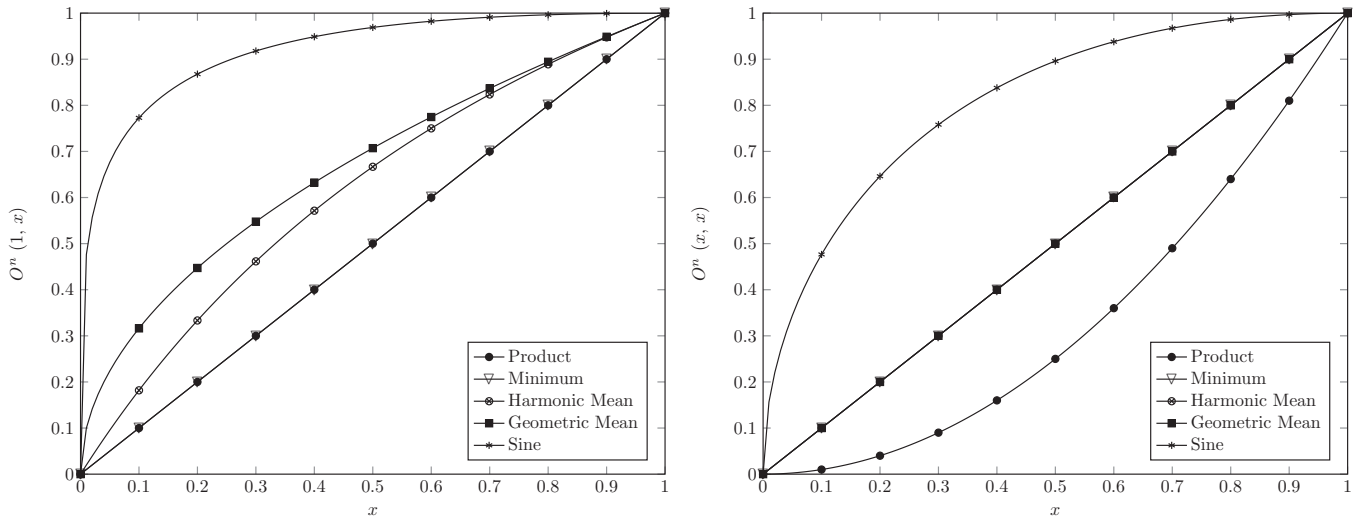
$$O(x, \ldots, x) = x. \tag{30}$$

The reason is that the behavior of idempotent overlap functions is not affected by the number of antecedents. As we can observe, this property is satisfied by the minimum t-norm (Eq. (26)) and the harmonic (Eq. (27)) and geometric (Eq. (28)) means.

Fig. 1 (a) and (b) shows the previously mentioned differences in the behavior of the different overlap functions (we consider the two dimensional case, $n = 2$, to easily visualize their behavior). In Fig. 1(a), we can observe the values returned by each overlap function when aggregating a value with 1, whereas Fig. 1(b) depicts the returned values when aggregating a value with itself. Having a look at Fig. 1(a) and (b), we can observe that the proposed n-dimensional overlap functions provide values with a higher variation than the product when aggregating small values. Nevertheless, both figures reveal a huge difference between the SIN and the rest of overlap functions, since the value returned by the SIN is greater than the input arguments when aggregating a value with itself (Fig. 1(b)). We will experimentally show that this behavior may not be desirable in this framework, as it may produce a loss of discrimination power in the FRBCS. However, we have included this function aiming at obtaining a general overview of n-dimensional overlap functions and showing their behavior based on results.

5.2. Applying n-dimensional overlap functions in FRBCSs

One of the objectives of this work is to extend the usage of n-dimensional overlap functions introduced in [15] to other FRBCSs aiming at improving the performance when decomposition strategies are used. In this manner, we apply these functions to model the conjunction in fuzzy rules. As shown in the previous section, the aggregation of small values when using the product t-norm produces values with a low variation that tend quickly to 0. Moreover, when we consider FRBCSs where the number of antecedents can vary depending on the rule, this effect is even more accentuated in those rules with a higher number of antecedents. These factors have a different influence in baseline FRBCSs, as output values are not used beyond the classification. However, this is an undesirable circumstance when using decomposition strategies, since the knowledge acquired in the base classifiers is partially lost, producing a negative impact on the aggregation phase of these strategies.



(a) Considering $(1, x)$ as input

(b) Considering (x, x) as input

Fig. 1. Values returned by the different overlap functions.

In order to minimize the loss of knowledge and to obtain more suitable confidences when using decomposition strategies and FRBCSs, we propose to use n-dimensional overlap functions to model the conjunction in fuzzy rules. In this manner, the greater variation of the outputs of these functions and the fact that they are independent of the number of input arguments make confidences more suitable for the aggregation phase.

With the aim of studying the performance of n-dimensional overlap functions in multiple types of FRBCSs and obtaining the broadest possible overview, we have considered four different FRBCSs. The following is a detailed description of the application of overlap functions in each classifier.

5.2.1. Introducing n*-dimensional overlap functions in the FRM of CHI*

In this classifier, overlap functions replace the t-norm used in the matching and association degrees computation (Eqs. (3) and (4), respectively), in the same way as it was done in [15] :

- *Matching degree*:

$$\mu_{A_j}(x_p) = O\big(\mu_{A_{j1}}(x_{p1}), \ldots, \mu_{A_{jn}}(x_{pn})\big) \tag{31}$$

- *Association degree*:

$$b_j(x_p) = O\big(\mu_{A_j}(x_p), RW_j\big) \tag{32}$$

As we described in Section 3.2, CHI algorithm does not perform any feature selection process and thus, all rules have exactly the same number of antecedents. Moreover, during the learning stage a rule is generated for each single instance in the training set, so the number of rules is the same when we use any overlap function. Therefore, we can observe that the usage of overlap functions does not have any major effect in the learning process, since neither the matching degree nor the association degree are considered for the generation of rules, except for rule weights computation (Eq. (2)).

5.2.2. Introducing n*-dimensional overlap functions in the FRM of SLAVE*

The computation of the association and adaptation degrees (Eqs. (9) and (10)) is carried out by an overlap function, instead of the product:

- *Adaptation degree*:

$$U_j(x_p, A_j) = O\big(Poss(A_{j1}|x_{p1}), Poss(A_{j2}|x_{p2}), \ldots, Poss(A_{jn}|x_{pn_j})\big) \tag{33}$$

- *Association degree*:

$$b_j(x_p) = O\big(U_j(x_p, A_j), RW_j\big) \tag{34}$$

This algorithm (Section 3.3) carries out a feature selection process that is embedded into the learning stage, and therefore the number of antecedents may vary depending on the rule. In this case, the usage of overlap functions affects the learning process, and hence the number of antecedents and rules generated when using different overlap functions varies. The reason is that SLAVE makes use of the association and adaptation degrees during the feature and rule selection processes. Consequently, overlap functions are not only involved in the inference process but also have a direct influence on the rule base generated in the learning phase.

5.2.3. Introducing n*-dimensional overlap functions in the FRM of FURIA*

Overlap functions are applied in the same manner as in CHI (Eqs. (31) and (32)), i.e., in the matching and association degrees (Eq. (3) and (4)). However, in this case the values to be aggregated are those returned by the different trapezoidal membership functions, instead of triangular ones.

FURIA (Section 3.4) learns all rules using RIPPER algorithm before fuzzifying them. This means that neither the number of antecedents nor the number of rules generated in the learning process depend on the overlap function used. Indeed, FURIA applies fuzzy sets theory after learning all rules in order to replace the interval of each antecedent by a trapezoidal membership function, but this fuzzification is performed only considering the purity of the rule (Eq. (14)), which is not computed using any t-norm. Thus, we can observe that overlap functions are not involved in the learning process of FURIA, except for the computation of rule weights (as in CHI).

5.2.4. Introducing n*-dimensional overlap functions in the FRM of FARC-HD*

The adaptation carried out in FARC-HD is the same as that performed in CHI (Eqs. (31) and (32)). As in SLAVE, overlap functions are involved in all stages of the learning process of FARC-HD, since it makes use of both the matching and association degrees to extract the fuzzy rules and to perform the feature and rule selection processes. As we described in Section 3.5, this algorithm performs a lateral tuning of linguistic labels in order to improve the classification accuracy. Since the prediction is made using the matching and association degrees (Eq. (3)–(6)), the usage of overlap functions also affects the previously mentioned lateral tuning. This adjustment of membership functions helps FARC-HD to increase the benefits of overlap functions.

6. Experimental framework

This section is aimed at presenting the experimental framework setup used to carry out the experiments in Section 7, which is the same as that considered in [15]. First, we show the features of the datasets selected for the experimental study (Section 6.1). Next, the parameter setup considered for each method is described (Section 6.2). Finally, we introduce the performance measures and the statistical tests that are necessary to assess whether significant differences exist among the results obtained (Section 6.3).

6.1. Datasets

In order to test the performance of the different methods, we have considered twenty datasets selected from the KEEL dataset repository [3]. In Table 2, we find a summary of the features of all datasets, indicating for each one the number of examples (#Ex.), number of attributes (#Atts.), number of numerical (#Num.) and nominal (#Nom.) attributes, and the number of classes (#Class.).

All the experiments have been carried out using a *5-fold stratified cross-validation model*, i.e., we randomly split the dataset into five partitions of data, each one containing 20% of the examples, and we employed a combination of four of them (80%) to train the system and the remaining one to test it. Additionally, in each partition we consider three different seeds for the execution of the methods. Therefore, the result for each dataset is computed as the average of the five partitions using the three seeds in each one. In order to correct the dataset shift, that is, when the training data and the test data do not follow the same distribution, we will use a recently published partitioning procedure called *Distribution Optimally Balanced Cross Validation* [41], instead of the commonly used cross-validation.

Table 2
Summary of the features of the datasets used in the experimental study.

Id.	Dataset	#Ex.	#Atts.	#Num.	#Nom.	#Class.
aut	autos	159	25	15	10	6
bal	balance	625	4	4	0	3
cle	cleveland	297	13	13	0	5
con	contraceptive	1473	9	6	3	3
eco	ecoli	336	7	7	0	8
gla	glass	214	9	9	0	7
hay	hayes-roth	132	4	4	0	3
iri	iris	150	4	4	0	3
new	newthyroid	215	5	5	0	3
pag	pageblocks	548	10	10	0	5
pen	penbased	1100	16	16	0	10
sat	satimage	643	36	36	0	7
seg	segment	2310	19	19	0	7
shu	shuttle	2175	9	9	0	5
tae	tae	151	5	3	2	3
thy	thyroid	720	21	21	0	3
veh	vehicle	846	18	18	0	4
vow	vowel	990	13	13	0	11
win	wine	178	13	13	0	3
yea	yeast	1484	8	8	0	10

Table 3
Setup of the methods parameters.

Algorithm	Parameters
CHI	Num. of linguistic labels per variable: 3 Rule weight: certainty factor
SLAVE	Num. of linguistic labels per variable: 5 Number of individuals: 100 Mutation probability: 0.01 Max. iterations without change: 500
FURIA	Num. of optimizations: 2 Num. of folds: 3
FARC-HD	Num. of linguistic labels per variable: 5 Minimum support: 0.05 Minimum confidence: 0.8 Maximum depth: 3 Parameter k: 2 Evaluations: 20,000 Number of individuals: 50 α parameter: 0.02 Bits per gen: 30 Rule weight: certainty factor

6.2. Methods setup

Table 3 shows the configuration and parameters that we have considered for each FRBCS. The source code of all baseline classifiers was obtained from KEEL software [3]. The selected values are common for all problems, and they were selected according to the recommendation of the authors of each algorithm. Even though the tuning of parameters for each method on each particular problem could lead to better results, we preferred to maintain a baseline performance on each method as the basis for comparison, since we are not comparing algorithms among them.

6.3. Performance measures and statistical tests

In this paper we have used the most common metric to test the performance of different methods, that is, the *accuracy rate*, which measures the percentage of correctly classified examples related to the total number of examples. However, we cannot extract well justified conclusions based only on the accuracy. For this reason, we apply some non-parametric tests [24] with the aim of providing statistical support to our results. Specifically, we use the Wilcoxon signed-ranks test [49] to perform pairwise comparisons, the Aligned Friedman test [27] to detect statistical differences among a group of methods, and the Holm post-hoc test [28] to find the algorithms that reject the null hypothesis of equivalence against the selected control method. A complete description of these tests can be found on the website: http://sci2s.ugr.es/sicidm/.

In addition to the previously mentioned performance measures, we also want to study the impact of overlap functions on the rule base. With this aim, we compute the average number of rules and antecedents per rule for each overlap function in both OVO and OVA models and in all baseline FRBCSs considered in this work. In the case of decomposition strategies, the average of all base classifiers is computed.

7. Experimental study

In this section, we study the results obtained by each method carrying out an analysis composed of four stages:

1. We test the performance of the different n-dimensional overlap functions when applying OVO and OVA models in all the FRBCSs considered in this paper (Section 7.1).
2. We study the impact of n-dimensional overlap functions on the rule base (Section 7.2).
3. We check whether the problems of WV with the confidences of FARC-HD are also present in the rest of FRBCSs considered in this paper comparing the original WV against WinWV (Section 7.3).
4. We discuss the results obtained in all previous points as a whole and we explain the reasons for the different behaviors in comparison with that obtained in FARC-HD (Section 7.4).

7.1. Analysis of the performance of n-dimensional overlap functions

Table 4 shows the average accuracy rate obtained in testing by the different FRBCSs (CHI, SLAVE, FURIA, and FARC-HD). As we can observe, we present the results obtained by each baseline FRBCS along with OVA scheme and with the five aggregation strategies of OVO model (ND, VOTE, LVPC, WV, and WinWV). We show the performance of the five overlap functions (PROD, MIN, HM, GM, and SIN) for each method, where the result of the best overlap function is highlighted in bold-face. These results are

Table 4
Average accuracy rate obtained in testing by each method.

	CHI					SLAVE					FURIA					FARC-HD				
	PROD	MIN	HM	GM	SIN	PROD	MIN	HM	GM	SIN	PROD	MIN	HM	GM	SIN	PROD	MIN	HM	GM	SIN
Baseline	**75.07**	71.20	67.41	66.92	65.74	**76.71**	74.37	74.31	74.54	73.08	**80.56**	80.55	80.46	80.49	80.12	**80.37**	80.17	80.11	79.89	79.98
OVA	**73.76**	67.39	64.20	63.36	61.93	**69.91**	69.13	68.97	68.47	64.85	80.39	80.38	80.40	80.36	**80.41**	79.92	80.27	**80.48**	80.13	79.97
OVO^{ND}	**77.10**	74.86	72.65	71.94	70.55	**77.41**	77.03	76.68	76.39	76.55	81.97	**81.98**	81.92	81.90	81.67	81.45	81.88	**82.18**	82.13	81.46
OVO^{VOTE}	**77.90**	75.55	73.72	73.05	71.98	**77.73**	77.34	77.17	76.72	77.06	82.37	**82.39**	82.37	82.34	82.12	81.52	82.03	**82.26**	82.25	81.71
OVO^{LVPC}	**77.93**	74.74	72.86	72.30	70.88	**71.72**	70.97	69.44	69.04	70.66	**82.52**	**82.52**	82.36	82.29	82.11	**79.77**	79.61	79.29	79.06	78.80
OVO^{WV}	**78.11**	75.31	73.42	72.75	71.17	**72.23**	71.63	69.73	69.33	71.79	**82.61**	82.58	82.45	82.38	82.20	80.19	80.16	**80.24**	80.05	78.96
OVO^{WinWV}	**78.19**	75.53	73.75	73.10	71.94	**76.07**	75.73	75.34	74.97	76.06	82.44	**82.45**	82.42	82.39	82.11	81.50	81.86	**81.93**	81.89	81.39

Table 5
Aligned Friedman and Holm tests to compare the different overlaps in FARC-HD, OVA and OVO.

	FARC-HD	OVA	OVO^{ND}	OVO^{VOTE}	OVO^{LVPC}	OVO^{WV}	OVO^{WinWV}
PROD	**43.80**	57.90 (0.128)	55.23 (0.327)	56.53 (0.269)	**37.90**	**42.38**	54.40 (0.747)
MIN	48.63 (0.967)	51.72 (0.282)	49.03 (0.708)	49.77 (0.672)	41.22 (0.717)	42.95 (1.000)	46.42 (1.000)
HM	50.22 (0.967)	**38.23**	**40.52**	**40.95**	54.05 (0.157)	43.90 (1.000)	**43.83**
GM	56.25 (0.699)	48.95 (0.282)	45.65 (0.708)	43.65 (0.768)	56.67 (0.122)	49.13 (1.000)	47.95 (1.000)
SIN	53.60 (0.856)	55.70 (0.170)	62.08 (0.075)	61.60 (0.097)	62.65 (0.028)	74.15 (0.002)	59.90 (0.319)

obtained by computing the average accuracy rate of each method in all datasets. The result of each dataset is computed as the average accuracy rate of the five partitions over the three different seeds.

Additionally, in order to detect significant differences among the results of each overlap function in a given FRBCS, we carry out the Aligned Friedman test and the Holm post-hoc test, whose results are shown in Tables 5–8. The results of these tests are grouped in columns according to the method used to perform the comparison and in rows according to the overlap function considered. The first column corresponds to the baseline FRBCS execution applying each overlap function, whereas the second one shows the different overlap functions over OVA model. The rest of columns correspond to all OVO aggregation strategies considered in this paper (ND, VOTE, LVPC, WV and WinWV). The value of each cell corresponds to the rank obtained with the Aligned Friedman test when comparing the different overlap functions for each method (that is, an Aligned Friedman test is carried out for each group of methods in a column). The value shown in brackets indicates the adjusted p-Value obtained by the Holm post-hoc test using as control method the one obtaining the lowest rank in the same column, which is shown in bold-face. The adjusted p-Value is underlined when there are statistical differences ($\alpha = 0.1$ considering the ratio between datasets and algorithms).

Next, we explain the behavior of n-dimensional overlap functions in each baseline FRBCS, as well as when decomposition strategies are applied on them. We start describing the results obtained with FARC-HD, since n-dimensional overlap functions were first introduced in this FRBCS and we want to analyze the existing differences between the results obtained in this method with those in the remaining ones.

- *FARC-HD*
  - *Baseline*: as we can observe in Table 4, the five overlap functions considered in this paper obtain a similar performance. This is confirmed by the Aligned Friedman test shown in Table 5, since there are no statistical differences among them when executing the baseline FARC-HD algorithm. This means that FARC-HD is able to maintain the necessary classification accuracy when using overlap functions. The reason is that these functions are involved in all stages of the learning process (Section 5.2.4) and the generated rules are general enough to retain the discrimination capability.
  - *OVO and OVA models*: leaving the SIN aside, Tables 4 and 5 show that the greater the overlap function is, the better the results obtained are (although the GM is greater than the HM, both of them have a similar behavior). The problem with the SIN is that the value returned can be greater than all input values, which may not be a desirable behavior for an inference system because part of the discrimination capability is lost. Therefore, we observe that the best overlap functions in almost all cases are those returning the highest values preserving the idempotency property (HM and GM). Although the geometric and harmonic means return similar values, the latter one tends to obtain better results but without statistical differences. FARC-HD is able to take advantage of the confidences provided by these functions, since the classification accuracy is maintained when using overlap functions in the baseline model. Nevertheless, when LVPC and WV aggregations are used, the behavior of overlap functions changes due to the factors that will be described in Section 7.3. For this reason, a new aggregation strategy (WinWV) was presented in [15], which solved the problems of LVPC and WV with the confidences given by FARC-HD. This new aggregation method along with the problems of LVPC and WV when using FARC-HD are described in Section 7.3.
- *SLAVE*
  - *Baseline*: looking at the accuracy rate (Table 4), we can observe that the product performs much better than the remaining overlap functions. This situation is confirmed in the statistical tests (Table 6), where there are significant differences in favor of the usage of the product when executing the baseline SLAVE. This classifier uses a different rule structure from that used in FARC-HD, which requires the greater discrimination capability provided by the product. There are some important differences between the rule structure of SLAVE and FARC-HD:

Table 6
Aligned Friedman and Holm tests to compare the different overlaps in SLAVE, OVA and OVO.

	SLAVE	OVA	OVO^{ND}	OVO^{VOTE}	OVO^{LVPC}	OVO^{WV}	OVO^{WinWV}
PROD	**21.47**	**37.02**	**33.63**	**32.40**	**30.95**	**30.55**	**40.42**
MIN	52.52 (0.001)	42.23 (1.000)	44.08 (0.255)	46.85 (0.115)	44.35 (0.144)	41.70 (0.224)	44.20 (0.681)
HM	56.63 (0.000)	41.05 (1.000)	53.90 (0.054)	51.10 (0.083)	61.27 (0.003)	66.10 (0.000)	53.48 (0.465)
GM	52.08 (0.001)	53.90 (0.198)	64.23 (0.003)	65.30 (0.001)	66.65 (0.000)	68.65 (0.000)	63.80 (0.043)
SIN	69.80 (0.000)	78.30 (0.000)	56.67 (0.036)	56.85 (0.023)	49.27 (0.092)	45.50 (0.206)	50.60 (0.535)

Table 7
Aligned Friedman and Holm tests to compare the different overlaps in CHI, OVA and OVO.

	CHI	OVA	OVO^{ND}	OVO^{VOTE}	OVO^{LVPC}	OVO^{WV}	OVO^{WinWV}
PROD	**20.18**	**20.07**	**22.60**	**21.43**	**19.70**	**21.60**	**22.30**
MIN	28.00 (0.394)	35.92 (0.084)	33.02 (0.256)	33.65 (0.183)	35.45 (0.086)	32.63 (0.229)	34.50 (0.184)
HM	64.85 (0.000)	59.65 (0.000)	60.72 (0.000)	59.93 (0.000)	59.55 (0.000)	58.60 (0.000)	59.50 (0.000)
GM	67.38 (0.000)	65.57 (0.000)	63.55 (0.000)	65.45 (0.000)	63.68 (0.000)	64.03 (0.000)	64.58 (0.000)
SIN	72.10 (0.000)	71.27 (0.000)	72.60 (0.000)	72.05 (0.000)	74.13 (0.000)	75.65 (0.000)	71.63 (0.000)

1. The rules generated in SLAVE are more specific (with more antecedents) than in FARC-HD.
2. As shown in Section 3.5, FARC-HD performs a lateral tuning in order to adjust the membership functions of fuzzy sets. As we stated, this adjustment is performed applying overlap functions (Section 5.2.4), and hence the classifier accuracy optimization is carried out considering the overlap functions, increasing the benefits with respect to SLAVE.
3. The values of the antecedents in SLAVE are subsets of linguistic labels instead of single linguistic labels as in FARC-HD. For this reason, SLAVE needs to model the disjunction in fuzzy rules, which can cause a different effect when using overlap functions.

- *OVO model*: Table 4 shows that the accuracy obtained by all overlap functions when OVO model is considered is similar, although a decreasing trend can be observed as the overlap function increases. Looking at the statistical analysis in Table 6, we can observe that, even though accuracy rates are similar, there are statistical differences in favor of the product. However, it should be stressed that ranking differences between the product and the remaining overlap functions are reduced. This means that OVO takes advantage of the confidences returned by overlap functions, since the usage of these functions allows the performance of their respective base classifier to be raised in such a way that they obtain more similar results to that of the product. Therefore, the ratio of improvement of overlap functions in this model is greater than that of the product. The problem is that in this case the base classifiers do not provide enough classification accuracy (as it was shown in the baseline SLAVE) to obtain an improvement in OVO model when using overlap functions as in the case of FARC-HD.
- *OVA model*: as we can observe in Tables 4 and 6, contrary to the rest of the FRCBSs considered in this paper, the performance of this strategy is worse than that of the baseline SLAVE. This is due to the class imbalance problem that appears in this strategy and the inability of SLAVE to deal with this situation. Therefore, the behavior of overlap functions in this case is not representative in our framework.

- *CHI*
  - *CHI*: taking a look at Table 4, we observe that the greater the overlap function is, the worse the results obtained are. This situation is confirmed by the Aligned Friedman and Holm post-hoc tests (Table 7), where we find significant differences in favor of the product. The reason for this behavior in comparison with FARC-HD is that the rules generated by CHI are much more specific than those of FARC-HD, since the number of antecedents is always equal to the number of features and they are learned considering all classes at the same time (whereas FARC-HD generates the rules class by class). Consequently, CHI algorithm needs more discrimination capability than FARC-HD due to the fact that the generated fuzzy rules are closer among themselves. Therefore, the usage of the product t-norm produces a greater discrimination power and leads to obtaining better results, whereas overlap functions highly affect the decision boundaries.
  - *OVO and OVA models*: Table 4 shows that, contrary to FARC-HD, the usage of overlap functions in the baseline CHI algorithm implies a loss of accuracy. Although in SLAVE this problem appears as well, the loss of accuracy in CHI is too great to obtain benefits from the confidences provided by overlap functions, as it is confirmed in the Aligned Friedman tests shown in Table 7. As a consequence, the behavior of these functions in OVO and OVA is the same as that observed in the baseline CHI.
- *FURIA*
  - *Baseline*: Tables 4 and 8 show that the behavior of all the overlap functions is similar in FURIA, and hence this algorithm is able to maintain the classification accuracy when using these functions (except for the SIN). In this case, overlap functions are not involved in any of the learning stages of FURIA. This is because rules are generated by RIPPER algorithm and t-norms are not used in the subsequent fuzzification process. Thus, the rules generated when using different overlap functions will be the same. Furthermore, FURIA uses highly adjusted trapezoidal membership functions (whose adjustment

Table 8
Aligned Friedman and Holm tests to compare the different overlaps in FURIA, OVA and OVO.

	FURIA	OVA	OVO^{ND}	OVO^{VOTE}	OVO^{LVPC}	OVO^{WV}	OVO^{WinWV}
PROD	**40.17**	51.00 (1.000)	42.65 (0.871)	43.75 (1.000)	**38.25**	**34.65**	45.60 (1.000)
MIN	43.10 (0.750)	51.75 (1.000)	**40.90**	**42.77**	38.75 (0.956)	37.02 (0.796)	**41.60**
HM	50.90 (0.727)	50.00 (1.000)	48.05 (0.871)	44.50 (1.000)	52.53 (0.239)	50.78 (0.158)	45.22 (1.000)
GM	48.97 (0.727)	54.73 (1.000)	53.80 (0.479)	50.52 (1.000)	57.35 (0.112)	57.25 (0.041)	50.25 (1.000)
SIN	69.35 (0.006)	**45.03**	67.10 (0.018)	70.95 (0.008)	65.62 (0.011)	72.80 (0.000)	69.83 (0.008)

is not performed using t-norms) which provide high membership degrees, and hence the differences among the values returned by different overlap functions when aggregating large values are lower (Fig. 1(a) and (b)).

- *OVO and OVA models*: as we can observe in Tables 4 and 8, both OVO and OVA models provide a similar performance when using different overlap functions. Even though FURIA maintains the classification accuracy when using overlap functions, the confidences provided by overlap functions are very similar due to the highly adjusted trapezoidal membership functions. The exception to this situation is when using the SIN overlap function due to the same reasons as those explained in the case of FARC-HD. In the same manner, LVPC and WV aggregation methods present different behaviors, which will be described in Section 7.3.

Summarizing, we can observe that the benefits of overlap functions are dependent on the learning process and the rule structure of each classifier. Therefore, the classifiers that are able to take advantage of overlap functions may be those which include these functions in their learning algorithms and have rules general enough to preserve the discrimination capability, maintaining the necessary classification accuracy. As we have shown, even though the confidences provided by overlap functions are more suitable for the aggregation performed in decomposition strategies, when the base classifiers do not provide enough classification accuracy these strategies do not obtain an improvement when using overlap functions.

Regarding decomposition strategies, Table 4 show their effectiveness when using FRBCSs, improving their performance in most of cases. However, the performance of OVA model can be affected by the increase in the imbalance ratio produced by the division performed in this strategy [20], as it occurs in SLAVE. We should keep in mind that in OVA each base classifier has to distinguish one of the classes from all others, and hence the proportion of instances of that class with respect to the rest of classes will be probably much smaller, particularly in datasets with a high number of classes (even if the original dataset is balanced).

7.2. *Impact of* n*-dimensional overlap functions on the rule base*

This subsection is aimed at showing the impact of *n*-dimensional overlap functions on the rule base. Table 9 presents the average number of rules and antecedents per rule for each baseline FRBCS and for OVA and OVO models (using as base classifier the same FRBCS). These averages are computed in the same manner as in Table 4, that is, by computing the average of each method in all datasets. The result of each dataset is computed as the average of the five partitions over the three different seeds.

Next, we analyze the effect of *n*-dimensional overlap functions on the rule base of each FRBCS:

- *FARC-HD:* Table 9 shows that the usage of a greater overlap function implies a growing trend in the number of rules. A higher number of rules is needed in order to maintain or improve the discrimination capability, since the aggregation of the membership degrees returns larger values. On the other side, the number of antecedents is similar in all overlap functions, even though there is an upward trend when using greater overlap functions.
- *SLAVE:* As we can observe in Table 9 and contrary to FARC-HD, when we use greater overlap functions the number of rules decreases, whereas the number of antecedents increases. This behavior can be produced by two factors:
  1. Due to the fact that SLAVE uses disjunctions in their rules, the usage of a greater overlap function may imply an increase in the number of linguistic labels in the antecedents, instead of implying an increase in the number of rules as in FARC-HD.
  2. Since the rules generated in SLAVE are more specific than those generated in FARC-HD, the discrimination capability can be partially lost when using greater overlap functions, requiring the usage of more antecedents in the rules in order to maintain the necessary discrimination power.
- *CHI:* Since this algorithm does not perform any feature selection process, all rules will have exactly the same number of antecedents (equal to the number of features of the problem), and thus the usage of overlap functions does not alter the number of antecedents of the rules, as it can be observed in Table 9. Furthermore, CHI algorithm generates a new rule for each example, and consequently overlap functions do not have any effect in the number of rules generated. Another consequence of generating a new rule for each example is that the number of rules is notably greater than in the rest of methods. It should

Table 9
Average number of rules and antecedents.

		Avg. rules					Avg. antecedents				
		PROD	MIN	HM	GM	SIN	PROD	MIN	HM	GM	SIN
CHI	Baseline	170.43	170.43	170.43	170.43	170.43	12.40	12.40	12.40	12.40	12.40
	OVA	155.49	155.49	155.49	155.49	155.49	12.40	12.40	12.40	12.40	12.40
	OVO	86.94	86.94	86.94	86.94	86.94	12.40	12.40	12.40	12.40	12.40
SLAVE	Baseline	18.47	17.15	16.37	16.75	17.38	4.33	3.64	6.71	6.75	7.32
	OVA	3.78	3.61	3.45	3.40	3.56	2.36	2.11	3.45	3.38	3.85
	OVO	4.14	4.05	3.95	3.94	4.06	2.51	2.22	3.74	3.84	4.47
FURIA	Baseline	16.54	16.54	16.54	16.54	16.54	2.76	2.76	2.76	2.76	2.76
	OVA	7.95	7.95	7.95	7.95	7.95	2.05	2.05	2.05	2.05	2.05
	OVO	4.50	4.50	4.50	4.50	4.50	1.58	1.58	1.58	1.58	1.58
FARC-HD	Baseline	32.67	35.70	40.15	41.11	46.30	2.34	2.38	2.44	2.44	2.47
	OVA	13.03	14.26	16.09	16.64	18.35	1.76	1.79	1.84	1.84	1.86
	OVO	8.55	9.72	11.28	11.72	12.58	1.61	1.63	1.66	1.66	1.69

Table 10
Wilcoxon test to compare WinWV and WV.

FRCBS	WinWV vs. WV	R+	R−	*p*-Value	Hypothesis
	PROD	203.00	7.00	0.000	Rejected for WinWV at 95%
	MIN	189.00	21.00	0.002	Rejected for WinWV at 95%
FARC-HD	HM	195.50	14.50	0.001	Rejected for WinWV at 95%
	GM	196.50	13.50	0.001	Rejected for WinWV at 95%
	SIN	204.50	5.50	0.000	Rejected for WinWV at 95%
	PROD	197.50	12.50	0.001	Rejected for WinWV at 95%
	MIN	208.00	2.00	0.000	Rejected for WinWV at 95%
SLAVE	HM	204.50	5.50	0.000	Rejected for WinWV at 95%
	GM	196.00	14.00	0.001	Rejected for WinWV at 95%
	SIN	203.50	6.50	0.000	Rejected for WinWV at 95%
	PROD	107.00	103.00	0.981	Not rejected
	MIN	92.50	117.50	0.776	Not rejected
CHI	HM	106.00	104.00	0.959	Not rejected
	GM	109.50	100.50	0.910	Not rejected
	SIN	164.00	46.00	0.044	Rejected for WinWV at 95%
	PROD	50.00	160.00	0.044	Rejected for WV at 95%
	MIN	67.00	143.00	0.179	Not rejected
FURIA	HM	97.00	113.00	0.836	Not rejected
	GM	123.50	86.50	0.532	Not rejected
	SIN	87.00	123.00	0.469	Not rejected

be noted, however, that the number of rules is usually considerably lower than the number of examples, since multiple rules are removed due to conflicts.

- *FURIA:* As in CHI, overlap functions are not involved in the learning process of FURIA, and hence the rules generated are the same for all overlap functions, as it can be observed in Table 9.

With respect to the comparison between baseline and decomposition strategies, Table 9 clearly shows that the rule base becomes simpler when decomposition strategies are applied. This is because these strategies divide the original problem into easier-to-solve binary sub-problems, needing a lower number of rules and antecedents to solve each sub-problem. In the same manner, the rule base of the classifiers in OVO will be simpler than in OVA, since OVO scheme considers only the examples of two classes while OVA takes into account all examples in the training set. However, when OVA is applied with SLAVE, we observe that there are less rules and antecedents than in OVO. This is caused by the inability of this algorithm to deal with the increase in the imbalance ratio produced by the division performed in OVA (as mentioned in Section 7.1).

7.3. WinWV

As we showed in [15] and Section 7.1, WV and LVPC are severely affected by the poor quality of the confidences of the non-predicted classes provided by FARC-HD, which is accentuated in LVPC due to the difficulty in modeling the conflict and ignorance terms. We focused on solving the problems of WV with the confidence of the non-predicted class due to the fact that if the conflict and ignorance terms were not considered in LVPC, the original WV would be recovered [15]. In order to solve this problem, WinWV was proposed (described in Section 4.1).

In this section, we check whether this situation is also present in the remaining FRBCSs. To do so, we carry out a number of pair-wise comparisons using the Wilcoxon signed-ranks test to confront the proposed aggregation method and the original WV, considering all FRBCSs used in this paper (CHI, SLAVE, FURIA, and FARC-HD) and the five different overlap functions. Table 10 shows the results of these comparisons, where R+ and R- indicate the ranks obtained by WinWV and WV, respectively.

As we can observe, WinWV aggregation strategy statistically outperforms the original WV method with all overlap functions in the case of SLAVE and FARC-HD. On the contrary, when considering CHI and FURIA algorithms there are no statistical differences between both aggregation methods in almost all cases. The reason is that in these two algorithms the confidence of the non-predicted class does not equally affect the aggregation in WV, since they are likely to be equal to 0. In the case of FURIA, this is due to the highly adjusted trapezoidal membership functions, whereas in CHI the reason is the high number of antecedents in its rules. The exception appears when overlap functions that are also t-norms are applied on FURIA, where WV performs better than WinWV. The confidences of the non-predicted class provided by FURIA are usually large when they are higher than 0 due to the highly adjusted trapezoidal membership functions. As a consequence, the usage of overlap functions makes the confidences of the non-predicted class to be increased much more quickly than those of the predicted one, and hence both confidences become more similar. Consequently, this algorithm loses discrimination capability and obtains worse results with WinWV.

7.4. Discussion

After analyzing the performance of n-dimensional overlap functions and their impact on the rule base, we have shown that the results obtained depend on each FRBCS. Additionally, the experimental study shows that the problems of WV with the

confidences of FARC-HD are not present in all FRBCSs. For this reason, in this section we summarize and discuss all the previously mentioned points:

- *Performance of n-dimensional overlap functions*
  Even though the confidences provided by *n*-dimensional overlap functions are more suitable for the aggregation phase, decomposition strategies are not able to take advantage of these confidences if the base classifiers do not maintain the necessary classification accuracy when using these types of functions. This fact implies that the benefits obtained from the usage of *n*-dimensional overlap functions are strongly dependent on the learning algorithm and the rule structure of each FRBCS.
  In the case of FARC-HD, the baseline algorithm is able to maintain enough classification accuracy allowing decomposition strategies to take advantage of the confidences provided by overlap functions. In SLAVE, despite the fact that baseline classifiers do not provide enough accuracy to obtain an improvement with respect to the product, the confidences obtained from overlap functions allow one to reduce these differences. On the other side, the baseline CHI algorithm dramatically loses discrimination capability when using overlap functions, and hence decomposition techniques are not able to exploit these functions. Finally, FURIA provides a similar performance with all overlap functions, since the confidences returned by this algorithm are too high to obtain an improvement from these functions.
- *Effect of n-dimensional overlap functions on the rule base*
  The usage of *n*-dimensional overlap functions not only affects the model performance, but also the rule bases. In the case of FURIA and CHI the size of the rule base is exactly the same with all overlap functions, since they are not involved in the rules generation process. However, in those algorithms where overlap functions are involved in the learning process (FARC-HD and SLAVE), the rule base is different depending on the overlap function used. In FARC-HD, the usage of a greater overlap function implies an increase in the number of rules, whereas the number of antecedents remains similar with all of them. Regarding SLAVE, greater overlap functions produce less rules but with greater number of antecedents.
  In all cases, the rule base of each classifier becomes simpler when decomposition strategies are used. Likewise, the rule bases in OVO are simpler than in OVA. The exception is the case of OVA with SLAVE, where the rule base is even simpler than in OVO due to the class imbalance produced by OVA scheme (described in Section 7.1).
- *WinWV*
  In the case of FARC-HD and SLAVE, WinWV performs much better than WV. However, the results obtained by WinWV and WV are similar when considering CHI and FURIA. The reason is that the confidences of the non-predicted class provided by these two algorithms are likely to be equal to 0.

All in all, we have shown that overlap functions improve the confidences of classifiers for the subsequent aggregation phase, but this improvement is only translated into a significant enhancement of the final performance if the baseline classifier is able to maintain the accuracy. In the rest of cases, the differences with respect to the product are reduced when decomposition strategies are used, which shows the benefits of overlap functions. However, a deeper analysis must be carried out in these cases in order to maintain the discrimination capability of the baseline classifiers while improving the confidences so that final accuracy could be boosted. At the same time, the rule base varies from one overlap function to another when these functions are involved in the learning process. Finally, in some FRBCSs the confidences obtained for the non-predicted class negatively affect the prediction in decomposition strategies, in which case WinWV is beneficial.

8. Conclusions

This work was motivated by the improvement found in FARC-HD when applying *n*-dimensional overlap functions and decomposition strategies. In this paper, we have carried out an exhaustive study that has allowed us to understand the influence of *n*-dimensional overlap functions in different FRBCSs.

In order to do so, we have studied whether the methodology presented in [15] improves the performance of four different FRBCSs (CHI, SLAVE, FURIA, and FARC-HD). As we have shown, the performance of overlap functions strongly depends on the learning process and rule structure of each classifier. Contrary to FARC-HD, CHI and SLAVE algorithms are not able to maintain the necessary discrimination capability when using overlap functions. Consequently, even though the confidences returned by overlap functions are more suitable for decomposition strategies, no improvement can be obtained from them. On the other side, FURIA is capable of preserving the classification accuracy when applying overlap functions, but the usage of highly adjusted trapezoidal membership functions implies that the membership degrees to be aggregated are likely to be 0 or close to 1. This produces small differences among the values returned by the different overlap functions, and hence they present similar behaviors in FURIA. In addition to the performance of overlap functions, we have analyzed their effect on the rule base of each FRBCS.

To sum up, after analyzing the behavior of overlap functions in four different FRBCSs, we can conclude that the performance of decomposition strategies will be significantly enhanced in those classifiers that involve the usage of overlap functions in their learning processes, maintaining the necessary discrimination capability and providing enough classification accuracy in the base classifiers.

There are several aspects that remain to be addressed in future works. Among them, the issue of non-competent classifiers [21] must be considered when working with OVO scheme. Furthermore, a more in depth study of the effect of decomposition strategies on the interpretability of FRBCSs should be carried out. Finally, the comparison and combination between decomposition-based techniques and preprocessing-based fuzzy ensembles, such as bagging [48], could also be studied. In this

latter case, we would make use of fuzzy techniques with the unique aim of enhancing the classification performance, which is a completely different perspective than the one considered in this work.

Acknowledgment

This work has been supported by the Spanish Ministry of Science and Technology under the Project TIN-2013-40765-P.

References

[1] A. Acilar, A. Arslan, A novel approach for designing adaptive fuzzy classifiers based on the combination of an artificial immune network and a memetic algorithm, Inf. Sci. 264 (2014) 158–181.
[2] J. Alcalá-Fdez, R. Alcalá, F. Herrera, A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning, IEEE Trans. Fuzzy Syst. 19 (5) (2011) 857–872.
[3] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Multiple-Valued Log. Soft Comput. 17 (2–3) (2011) 255–287.
[4] R. Aliev, W. Pedrycz, B. Guirimov, R. Aliev, U. Ilhan, M. Babagil, S. Mammadli, Type-2 fuzzy neural networks with fuzzy clustering and differential evolution optimization, Inf. Sci. 181 (9) (2011) 1591–1608.
[5] G. Beliakov, A. Pradera, T. Calvo, Aggregation Functions: A Guide for Practitioners, Volume 221 of Studies in Fuzziness and Soft Computing, Springer, Berlin, Heidelberg, 2007.
[6] V. Bolón-Canedo, N.S.-M. No, A. Alonso-Betanzos, An ensemble of filters and classifiers for microarray data classification, Pattern Recognit. 45 (1) (2012) 531–539.
[7] P. Bonissone, J.M. Cadenas, M.C. Garrido, R.A. Díaz-Valladares, A fuzzy random forest, Int. J. Approx. Reason. 51 (7) (2010) 729–747.
[8] L. Breiman, Bagging predictors, Mach. Learn. 24 (1996) 123–140.
[9] H. Bustince, J. Fernandez, R. Mesiar, J. Montero, R. Orduna, Overlap functions, Nonlinear Anal.: Theory Methods Appl. 72 (3–4) (2010) 1488–1499.
[10] H. Bustince, M. Pagola, R. Mesiar, E. Hullermeier, F. Herrera, Grouping, overlap, and generalized bientropic functions for fuzzy modeling of pairwise comparisons, IEEE Trans. Fuzzy Syst. 20 (3) (2012) 405–415.
[11] Y. Chen, N.R. Pal, I. Chung, An integrated mechanism for feature selection and fuzzy rule extraction for classification, IEEE Trans. Fuzzy Syst. 20 (4) (2012) 683–698.
[12] Z. Chi, H. Yan, T. Pham, Fuzzy Algorithms with Applications to Image Processing and Pattern Recognition, World Scientific, 1996.
[13] W.W. Cohen, 1995. Fast effective rule induction, in: Presented at the 12th International Conference on Machine learning, Lake Tahoe, CA, USA.
[14] O. Cordón, M.J. del Jesus, F. Herrera, A proposal on reasoning methods in fuzzy rule-based classification systems, Int. J. Approx. Reason. 20 (1) (1999) 21–45.
[15] M. Elkano, M. Galar, J. Sanz, A. Fernández, E. Barrenechea, F. Herrera, H. Bustince, Enhancing multi-class classification in FARC-HD fuzzy classifier: On the synergy between *n*-dimensional overlap functions and decomposition strategies, IEEE Trans. Fuzzy Syst. 23 (5) (2014) 1562–1580.
[16] A. Fernández, M. Calderón, E. Barrenechea, H. Bustince, F. Herrera, Solving mult-class problems with linguistic fuzzy rule based classification systems based on pairwise learning and preference relations, Fuzzy Sets Syst. 161 (23) (2010) 3064–3080.
[17] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1) (1997) 119–139.
[18] J. Friedman, Another approach to polychotomous classification Technical Report, Department of Statistics, Stanford University, 1996.
[19] J. Fürnkranz, Round robin ensembles, Intell. Data Anal. 7 (5) (2003) 385–403.
[20] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes, Pattern Recognit. 44 (8) (2011) 1761–1776.
[21] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, Dynamic classifier selection for one-vs-one strategy: avoiding non-competent classifiers, Pattern Recognit. 46 (12) (2013) 3412–3424.
[22] M. Galar, A. Fernández, E. Barrenechea, F. Herrera, Empowering difficult classes with a similarity-based aggregation in multi-class classification problems, Inf. Sci. 264 (2014) 135–157.
[23] M. Galar, J. Sanz, M. Pagola, H. Bustince, F. Herrera, A preliminary study on fingerprint classification using fuzzy rule-based classification systems, in: Proceedings of the 2014 IEEE World Congress on Computational Intelligence (IEEE WCCI 2014) – 2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2014), 2014.
[24] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180 (10) (2010) 2044–2064.
[25] A. González, R. Perez, Slave: a genetic learning system based on an iterative approach, IEEE Trans. Fuzzy Syst. 7 (2) (1999) 176–191.
[26] S. Ho, C. Hsieh, H. Chen, H. Huang, Interpretable gene expression classifier with an accurate and compact fuzzy rule base for microarray data analysis, Biosystems 85 (3) (2006) 165–176.
[27] J.L. Hodges, E.L. Lehmann, Ranks methods for combination of independent experiments in analysis of variance, Ann. Math. Stat. 33 (1962) 482–497.
[28] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. 6 (1979) 65–70.
[29] J. Hong, J. Min, U. Cho, S. Cho, Fingerprint classification using one-vs-all support vector machines dynamically ordered with Naïve Bayes classifiers, Pattern Recognit. 41 (2) (2008) 662–671.
[30] J. Hühn, E. Hüllermeier, FURIA: an algorithm for unordered fuzzy rule induction, Data Min. Knowl. Discov. 19 (3) (2009) 293–319.
[31] J.C. Huhn, E. Hullermeier, Fr3: A fuzzy rule learner for inducing reliable classifiers, IEEE Trans. Fuzzy Syst. 17 (1) (2009) 138–149.
[32] E. Hüllermeier, K. Brinker, Learning valued preference structures for solving classification problems, Fuzzy Sets Syst. 159 (18) (2008) 2337–2352.
[33] E. Hüllermeier, S. Vanderlooy, Combining predictions in pairwise classification: an optimal adaptive voting strategy and its relation to weighted voting, Pattern Recognit. 43 (1) (2010) 128–142.
[34] H. Ishibuchi, T. Nakashima, M. Nii, Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining, Springer-Verlag, 2004.
[35] H. Ishibuchi, Y. Nojima, Y. Jin, Fuzzy ensemble design through multi-objective fuzzy rule selection, Multi-Objective Machine Learning, Volume 16 of Studies in Computational Intelligence, Springer, Berlin, Heidelberg, 2006, pp. 507–530.
[36] H. Ishibuchi, T. Yamamoto, T. Nakashima, Hybridization of fuzzy GBML approaches for pattern classification problems, IEEE Trans. Syst. Man Cybern. B 35 (2) (2005) 359–365.
[37] A. Jurio, H. Bustince, M. Pagola, A. Pradera, R.R. Yager, Some properties of overlap and grouping functions and their application to image thresholding, Fuzzy Sets Syst. 229 (2013) 69–90.
[38] W. Ling, W. Lu, Fuzzy rules extraction based on output-interval clustering and support vector regression for forecasting, J. Intell. Fuzzy Syst. 27 (2014) 2563–2571.
[39] A. Lorena, A. Carvalho, J. Gama, A review on the combination of binary classifiers in multiclass problems, Artif. Intell. Rev. 30 (1–4) (2008) 19–37.
[40] P. Melin, J. Amezcua, F. Valdez, O. Castillo, A new neural network model based on the LVQ algorithm for multi-class classification of arrhythmias, Inf. Sci. 279 (2014) 483–497.
[41] J. Moreno-Torres, J. Saez, F. Herrera, Study on the impact of partition-induced dataset shift on *k*-fold cross-validation, IEEE Trans. Neural Netw. Learn. Syst. 23 (8) (2012) 1304–1312.
[42] D. Paternain, M. Pagola, J. Fernandez, R. Mesiar, G. Beliakov, H. Bustince, 2011. Brain MRI thresholding using incomparability and overlap functions, in: Proceedings of the 2011 11th International Conference on Intelligent Systems Design and Applications (ISDA), pp. 808–812.

[43] J.A. Sáez, M. Galar, J. Luengo, F. Herrera, Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition, Knowl. Inf. Syst. 38 (1) (2014) 179–206.
[44] J. Sanz, D. Bernardo, F. Herrera, H. Bustince, H. Hagras, A compact evolutionary interval-valued fuzzy rule-based classification system for the modeling and prediction of real-world financial applications with imbalanced data, IEEE Trans. Fuzzy Syst. 23 (4) (2014) 973–990.
[45] J. Sanz, A. Fernández, H. Bustince, F. Herrera, IVTURS: a linguistic fuzzy rule-based classification system based on a new interval-valued fuzzy reasoning method with tuning and rule selection, IEEE Trans. Fuzzy Syst. 21 (3) (2013) 399–411.
[46] J. Sanz, M. Galar, A. Jurio, A. Brugos, M. Pagola, H. Bustince, Medical diagnosis of cardiovascular diseases using an interval-valued fuzzy rule-based classification system, Appl. Soft Comput. 20 (2014) 103–111.
[47] R. Senge, E. Hüllermeier, Top-down induction of fuzzy pattern trees, IEEE Trans. Fuzzy Syst. 19 (2011) 241–252.
[48] K. Trawinski, O. Cordon, L. Sanchez, A. Quirin, A genetic fuzzy linguistic combination method for fuzzy rule-based multiclassifiers, IEEE Trans. Fuzzy Syst. 21 (5) (2013) 950–965.
[49] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics 1 (6) (1945) 80–83.

3. CHI-BD: un nuevo sistema de clasificación basado en reglas difusas para problemas de clasificación Big Data – *CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems*

La publicación asociada a esta parte es:

- M. Elkano, M. Galar, J. Sanz, H. Bustince "CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems", *Fuzzy Sets and Systems*, en prensa, 2017. DOI: 10.1016/j.fss.2017.07.003.
    - Estado: Publicado (en prensa).
    - Índice de Impacto (JCR 2016): 2,718.
        - Computer Science, Theory & Methods. Ranking 18/104 (Q1).
        - Mathematics, Applied. Ranking 10/255 (Q1).
        - Statistics & Probability. Ranking 8/124 (Q1).

Available online at www.sciencedirect.com

ScienceDirect

Fuzzy Sets and Systems ••• (••••) •••–•••

FUZZY sets and systems

www.elsevier.com/locate/fss

CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems

Mikel Elkano [a,b,*], Mikel Galar [a,b], Jose Sanz [a,b], Humberto Bustince [a,b]

[a] *Departamento de Automatica y Computacion, Universidad Publica de Navarra, Navarra, 31006, Spain*
[b] *Institute of Smart Cities, Universidad Publica de Navarra, Navarra, 31006, Spain*

Received 11 May 2016; received in revised form 21 February 2017; accepted 2 July 2017

Abstract

The previous Fuzzy Rule-Based Classification Systems (FRBCSs) for Big Data problems consist in concurrently learning multiple Chi et al. FRBCSs whose rule bases are then aggregated. The problem of this approach is that different models are obtained when varying the configuration of the cluster, becoming less accurate as more computing nodes are added. Our aim with this work is to design a new FRBCS for Big Data classification problems (CHI-BD) which is able to provide exactly the same model as the one that would be obtained by the original Chi et al. algorithm if it could be executed with this quantity of data. In order to do so, we take advantage of the suitability of the Chi et al. algorithm for the MapReduce paradigm, solving the problems of the previous approach, which lead us to obtain the same model (i.e., classification accuracy) regardless of the number of computing nodes considered.

Keywords: Fuzzy Rule-Based Classification Systems; Big Data; Hadoop; MapReduce; Imbalanced datasets

1. Introduction

The enormous growth of the information created, replicated, and consumed by society has made *Big Data* one of the most popular research lines. According to the study carried out by Gantz and Reinsel [1], the information available will be almost doubled every two years from 2012 to 2020. On this account, the need to process large amount of data within a reasonable period of time arises. This scenario is known as Big Data. Although there is a wide variety of definitions of the term Big Data [2–4], which in many cases might be confusing and inaccurate, in this work we consider as Big Data problems those situations where the amount of information to be processed exceeds the computing capacity of a single machine. Nevertheless, there is no universal minimum size of data above which a certain dataset is considered as a Big Data problem, since it depends on the task to be performed on such data. In

* Corresponding author.
E-mail addresses: mikel.elkano@unavarra.es (M. Elkano), mikel.galar@unavarra.es (M. Galar), joseantonio.sanz@unavarra.es (J. Sanz), bustince@unavarra.es (H. Bustince).

http://dx.doi.org/10.1016/j.fss.2017.07.003
0165-0114/

Big Data contexts, *distributed computing* is a commonly applied strategy which splits the original dataset into smaller subsets that are distributed along multiple nodes of a *cluster*. In this manner, each node processes only the split of data stored in it. The outputs generated in all the nodes are then aggregated to obtain the final result.

One of the most popular frameworks for Big Data that follows the previous methodology is *Apache Hadoop*. This framework provides a transparent distributed system by implementing the *MapReduce* model [5] over a distributed file system called *HDFS* (Hadoop Distributed File System). HDFS is responsible for distributing data among nodes, whereas MapReduce allows one to process this data exploiting the data locality (i.e., avoiding data transfers over the network by preferably processing local data). MapReduce is composed of two phases: *map* and *reduce*. The map phase processes an input split of data generating different key-value pairs. Then, the reduce phase aggregates the results obtained in the map stage by key. Apache Hadoop is intended to build distributed systems with *horizontal scalability* (more nodes with typical desktop architectures) instead of *vertical* one (less nodes but more powerful) [6].

Among the problems that can be found in a Big Data context, *classification* tasks are found. In supervised classification, a learning algorithm extracts knowledge from a dataset (containing previously labeled instances) with the aim of classifying future input patterns or instances. One of the most popular machine learning techniques are Fuzzy Rule-Based Classification Systems (FRBCSs) [7]. These tools have been applied in numerous real applications, including bioinformatics [8,9], medical problems [10], anomaly intrusion detection [11], financial applications [12], and image processing [13], among others. However, the standard learning techniques rarely scale up to a distributed environment, and thus they are not able to address large classification problems in a reasonable time. Therefore, these techniques need to be adapted or redesigned in order to be applied in a distributed system.

In this framework, López et al. [14] presented an adaptation of the original Chi algorithm [15] to Big Data called Chi-FRBCS-BigDataCS. This method basically learns multiple fuzzy rule bases applying the Chi et al. learning algorithm in each mapper and aggregates all the rule bases in the reducer. However, this approach is not scalable in terms of classification performance, since the rule base generated in each mapper is learned using only the subset of instances processed by that specific mapper. As a consequence, the rule weights become highly dependent on the proportion and distribution of the classes in that subset. Therefore, the quality of the rule weights (and consequently the performance of this method) drops when increasing the number of mappers, since they are key in the inference process [16]. One should notice that the addition of new computing nodes (mappers) is mandatory as larger datasets are considered. In this work we refer to this method as CHI^{BD}_{Local} in order to show the difference with our proposal.

In this work, we aim to address the problems of the previous approach by designing a new distributed FRBCS for Big Data classification problems called CHI-BD. Our main objective is to be able to provide exactly the same model as that we would obtain if the FRBCS could be learned using a single computing node. That is, we want to obtain the same model (i.e., classification accuracy) regardless of the number of nodes used to learn the classifier. In order to do so, we also focus on the original Chi et al. [15] algorithm due to its suitability for the MapReduce paradigm, which was not exploited in [14]. Therefore, our main hypothesis is that considering all the data to build a global model should outperform the fusion of local models, besides being more robust to the configuration of the cluster. As we have done with CHI^{BD}_{Local}, along the paper we will refer to CHI-BD as CHI^{BD}_{Global} in order to show the main difference between both methods.

In order to achieve well-founded conclusions, we have developed an empirical study obtaining 20 binary datasets from 8 different multi-class problems available at the UCI repository [17]. Specifically, we will analyze the need for a scalable solution for the Chi et al. algorithm by comparing the original Chi et al. method [15] against CHI^{BD}_{Global}. Moreover, our new algorithm is compared with CHI^{BD}_{Local} [14] in terms of runtime and classification performance. Finally, the efficiency of our method is evaluated by the commonly used speedup, sizeup and scaleup measures [18,19].

The experimental study carried out shows that our new approach is able to address Big Data problems obtaining exactly the same results as the original Chi algorithm (if it could be executed in Big Data problems) regardless of the number of nodes used for the execution, unlike CHI^{BD}_{Local}. Additionally, the runtimes obtained show that CHI^{BD}_{Global} is considerably faster than CHI^{BD}_{Local} when dealing with large datasets. The entire source code of CHI^{BD}_{Global} is publicly available at GitHub (https://github.com/melkano/chi-bd) under the GNU General Public License.

This paper is organized as follows. Section 2 presents the preliminaries of this work with an introduction to Apache Hadoop, a brief description of the original Chi and CHI^{BD}_{Local} methods, and some related works. In Section 3, we present the new CHI^{BD}_{Global} classifier for Big Data problems. The experimental framework is given in Section 4 and the analysis of the results obtained is presented in Section 5. Finally, Section 6 concludes this paper.

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

2. Preliminaries

In this section we first introduce some basic concepts of the Apache Hadoop framework (Section 2.1). Next, we describe the original Chi classifier (Section 2.2) along with its adaptation to Big Data presented by López et al. in [14] (Section 2.3). Finally, we discuss some recent related works in Big Data classification (Section 2.4).

2.1. Apache Hadoop

Apache Hadoop (shortened to Hadoop) is probably the most popular framework for Big Data. The objective of this framework is to provide a transparent distributed system so that the user only has to focus on designing the processing of data. Hadoop is the open source alternative by Apache foundation to the Google's framework for Big Data, which is mainly based on two works: the *Google File System* (GFS) [20] and the *MapReduce* algorithm [5]. In this manner, the core of Hadoop consists of a distributed file system based on the GFS called *Hadoop Distributed File System* (HDFS) (storage part) and an implementation of MapReduce (processing part):

- *HDFS*: files are divided into physical blocks that are distributed among all the nodes. When processing a file, the code of the algorithm is transferred to all nodes in order to take advantage of data locality. This means that instead of moving the data, Hadoop moves the computation to the data. In this manner, a node will primarily process the blocks stored in it (although a block stored in another node can be processed as well), avoiding heavy data transfers.
- *MapReduce*: represents the processing part of Hadoop. This algorithm is composed of the following two phases.
  1. *Map phase*: input data is first divided into multiple logical splits that are associated with different physical blocks (preferably with local ones, in order to exploit the data locality). In this manner, each split will be processed by a single processing unit called *mapper* (although the user can assign multiple splits to a single mapper). Each node can execute multiple mappers concurrently. During the processing part carried out in a mapper, input data is transformed into <k, v>key-value pairs that are processed by the *map()* function (defined by the user), which is called for each pair. The result of this function is another <k′, v′>key-value pair that is part of the so-called *intermediate data*. Finally, this intermediate data is prepared to be sent to the reducers executing the following steps:
     (a) Sorting and Merging: outputs are sorted by key and all the values corresponding to the same key are merged in a list of values (<k′, list (v′)>).
     (b) Partitioning: a target reducer is selected for each key.
     (c) Shuffle: previous intermediate data (<k′, list (v′)>) are copied to the reducers.

     In addition to the *map()* function, there is another function called *cleanup()* that is invoked only once in the end of the execution of the mapper, that is, after all the input data have been processed by the *map()* function. The *cleanup()* function may return key-value pairs as well, which will be part of the intermediate data as in the case of the *map()* function. Generally, only one of these functions will be responsible for returning key-value pairs.
  2. *Reduce phase*: the reducer is responsible for aggregating the outputs of the mappers. In order to do so, once all the mappers have finished, all the key-value pairs that are received from them are sorted and merged by key. Then, the *reduce()* function (defined by the user) is called for every single key (k′), where all its values (list (v′)) are aggregated. Finally, the reducer provides the final result (v″) for each key.

  Fig. 1 depicts the data flow of the MapReduce algorithm. Besides the previous phases, there is an optimization for MapReduce jobs called *Combiner*. This function is locally executed on the output of the map phase (<k′, list (v′)>) and it is used as a local mini-reducer to lessen the intermediate data transferred to the reducers. In fact, in many cases the code of the combiner is the same as that of the reducer.

2.2. Fuzzy Rule-Based Classification Systems: Chi et al. algorithm

Fuzzy Rule-Based Classification Systems (FRBCSs) are one of the most popular tools used to solve classification problems. These systems provide an interpretable model by using human-readable rules composed of linguistic labels [7].

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

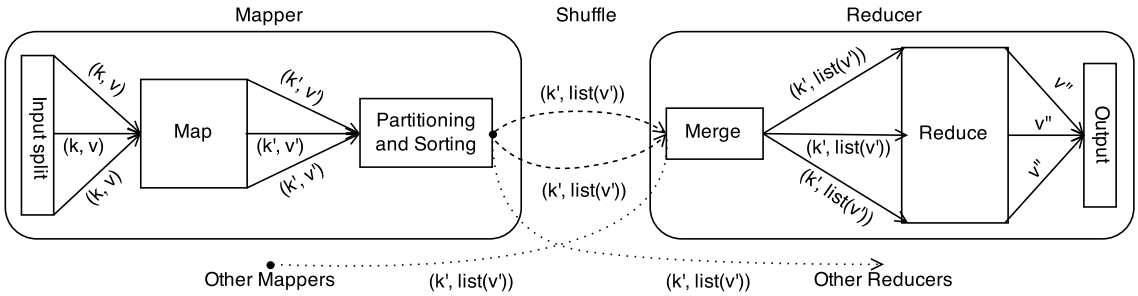


Fig. 1. MapReduce data flow.

The two main components of FRBCSs are the following ones.

1. *Knowledge base*: It is composed of both the rule base (RB) and the database, where the rules and the membership functions used to model the linguistic labels are stored, respectively.
2. *Fuzzy Reasoning Method (FRM)*: This is the mechanism used to classify examples using the information stored in the knowledge base.

In order to generate the knowledge base, a fuzzy rule learning algorithm is applied using a training set $\mathcal{D}_T$ composed of P labeled examples $x_p = (x_{p1}, \ldots, x_{pn})$ with $p \in \{1, \ldots, P\}$, where x_{pi} is the value of the i-th attribute ($i \in \{1, 2, \ldots, n\}$) of the p-th training example. Each example belongs to a class $y_p \in \mathbb{C} = \{C_1, C_2, \ldots, C_m\}$, where m is the number of classes of the problem.

Among the existing FRBCSs, we have considered the Chi et al. algorithm [15] because the learning process carried out by this FRBCS matches the working procedure of MapReduce, which allows us to fully exploit the potential of this framework, as we will see in Section 3. The rule structure used by this classifier is the following:

$$\text{Rule } R_j: \text{ If } x_1 \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_n \text{ is } A_{jn} \text{ then Class} = C_j \text{ with } RW_j \tag{1}$$

where R_j is the label of the j-th rule, $x = (x_1, \ldots, x_n)$ is an n-dimensional pattern vector that represents the example, A_{ji} is a linguistic label modeled by a triangular membership function, C_j is the class label and RW_j is the rule weight. One of the most common ways to compute the rule weight is the Penalized Certainty Factor (PFC) [16]:

$$RW_j = PCF = \frac{\sum_{x_p \in Class\ C_j} \mu_{A_j}(x_p) - \sum_{x_p \notin Class\ C_j} \mu_{A_j}(x_p)}{\sum_{p=1}^{P} \mu_{A_j}(x_p)} \tag{2}$$

where $\mu_{A_j}(x_p)$ is the matching degree of the example x_p with the antecedent part of the fuzzy rule R_j computed as follows.

$$\mu_{A_j}(x_p) = \prod_{i=1}^{n} \mu_{A_{ji}}(x_{pi}) \tag{3}$$

$\mu_{A_{ji}}(x_{pi})$ being the membership degree of the value x_{pi} to the fuzzy set A_{ji} of the rule R_j. Rule weights have recently been used to model the interaction of rules in a new FRM based on capacities [21].

In order to build the rule base, the following learning algorithm is applied.

1. *Construction of linguistic labels.* Fuzzy sets (linguistic labels) are built with the same triangular shape and equally distributed on the range of values.
2. *Generation of a fuzzy rule for each example*. A fuzzy rule is generated for each example x_p as follows.
   (a) The membership degrees of each value x_{pi} to all the different fuzzy sets of the i-th variable are computed.
   (b) For each variable, the linguistic label with the greatest membership degree is selected.

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

(c) The antecedent part is determined by the intersection of the selected linguistic labels and the consequent is the class label of the example (y_p). All rules will have exactly the same number of antecedents as variables in the problem (n).
(d) The rule weight is computed using Eq. (2).

As we can observe, rules with the same antecedent and different consequent part can be obtained after the rules generation process. In such case, only the one with the highest rule weight is kept. Rules with negative weight are removed from the rule base.

In order to classify a new example x_p, in this paper we consider the *winning rule* FRM [22], which is composed of the following steps.

1. *Matching degree*. The strength of activation of the antecedent part for all rules in the rule base with the example x_p is computed (Eq. (3)).
2. *Association degree*. The association degree of the example x_p with each rule in the rule base is computed.

$$b_j(x_p) = \mu_{A_j}(x_p) \cdot RW_j \tag{4}$$

3. *Classification*. The class of the rule with the highest association degree is predicted.

$$Class = arg \max_{c=1,\ldots,m} \left(\max_{R_j \in RB;\ C_j = c} b_j(x_p) \right) \tag{5}$$

2.3. *Chi-FRBCS-BigDataCS* (CHI^{BD}_{Local})

This approach was presented in [14] as an adaptation of the original Chi algorithm to Big Data imbalanced binary classification problems. In order to carry out this adaptation, the authors proposed the usage of Hadoop framework. This algorithm is composed of two steps:

1. *Generation of the rule bases*: an independent Chi classifier is learned in each mapper. In this manner, multiple Chi classifiers are concurrently obtained in different mappers. The learning process of each classifier is performed considering only the input split of instances associated with the mapper. Consequently, the rules and weights obtained in each classifier are generated using a subset of the training set. To this end, the map() function stores all the input examples in the main memory and the cleanup() function is where the Chi algorithm is applied to build the rule base. Once all classifiers have finished their learning phase, as many rule bases as mappers (classifiers) are obtained.
2. *Fusion of the rule bases*: all the rule bases generated in the previous phase are aggregated in a single reducer in order to provide the final rule base. In order to do so, all the rules are added to the final rule base. When two or more rules share the same antecedent part, only the one having the highest rule weight is kept in the final rule base.

Besides the adaptation to MapReduce, the authors applied cost-sensitive learning using the Penalized Cost-Sensitive Certainty Factor (PCF-CS) to compute the rule weight in order to deal with imbalanced datasets [23]. This is a modification of the previously presented PCF (Eq. (2)), which allows one to assign a different misclassification cost to each class. Hence, the rule weight is computed as

$$RW_j = PCF\text{-}CS = \frac{\sum_{x_p \in Class\ C_j} \mu_{A_j}(x_p) \cdot Cs(y_p) - \sum_{x_p \notin Class\ C_j} \mu_{A_j}(x_p) \cdot Cs(y_p)}{\sum_{p=1}^{P} \mu_{A_j}(x_p) \cdot Cs(y_p)}, \tag{6}$$

where $Cs(y_p)$ is the misclassification cost associated with the class y_p. Costs are defined as $Cs(min) = IR$ and $Cs(maj) = 1$, where min and maj are the minority and majority classes, respectively, and IR is the imbalanced ratio defined as P_{maj}/P_{min}, where P_{maj} and P_{min} are the number of instances belonging to the majority and minority

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

classes, respectively. This procedure allows one to better identify the instances of the minority class by assigning a higher weight to those instances.

Algorithms 1–3 show the pseudo-code (extracted from [14]) of this approach, while the entire data flow is described in Fig. 2. As we can observe, there are some limitations that make CHI^{BD}_{Local} not scalable in terms of classification performance. The main reason is that rule weights are computed using only a portion of the training data, and thus the quality of the rule weights obtained will be highly dependent on the proportion and distribution of each subset. This effect is increasingly accentuated as more nodes are added, since the number of examples in each subset becomes smaller. Consequently, the low quality of the rule weights affects the performance of this method as it is key factor in the inference process of fuzzy classifiers [16].

Algorithm 1 Mapper map() function for the CHI^{BD}_{Local} algorithm.

Procedure map (key, value)

Input: <*key*, *value*> pair, where *key* is the offset in bytes and *value* are the values of an instance.
Begin
1: *instance* ← *INSTANCE_REPRESENTATION* (*value*)
2: *instances* ← *instances.add* (*instance*) {*instances* will contain all instances in this mapper's split}
End

Algorithm 2 Mapper cleanup() function for the CHI^{BD}_{Local} algorithm.

Procedure cleanup ()

Output: <*key′*, *value′*> pair, where *key′* is any Long value and *value′* contains a RB.
Begin
1: *fuzzy_ChiBuilder.build* (*instances, Cs*) {*Cs* contains the misclassification cost of each class}
2: *ruleBase* ← *fuzzy_ChiBuilder.getRuleBase* ()
3: EMIT (*key′*, *ruleBase*)
End

Algorithm 3 Reducer reduce() function for the CHI^{BD}_{Local} algorithm.

Procedure reduce (key, values)

Input: <*key′*, *values*> pair, where *key′* is any Long value and *values* are the RBs generated by each mapper.
Output: <*key″*, *value″*> pair, where *key″* is a null value and *value″* is the final RB.
Begin
1: **while** *values.hasNext* () **do**
2: *ruleBase* ← *values.getValue* ()
3: **for** $i = 0$ **to** *ruleBase.size* () – 1 **do**
4: *rule* ← *ruleBase.get* (*i*)
5: **if** *finalRuleBase.contains* (*rule*) **then**
6: *finalRuleBase.takeRuleWithHighestWeight* (*rule*)
7: **else**
8: *finalRuleBase.add* (*rule*)
9: **end if**
10: **end for**
11: **end while**
12: EMIT (null, *finalRuleBase*)
End

2.4. Related work

Many distributed solutions based on MapReduce have been proposed for a wide variety of machine learning techniques in the last two years. Among them, we can find different approaches including sampling and prototype

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

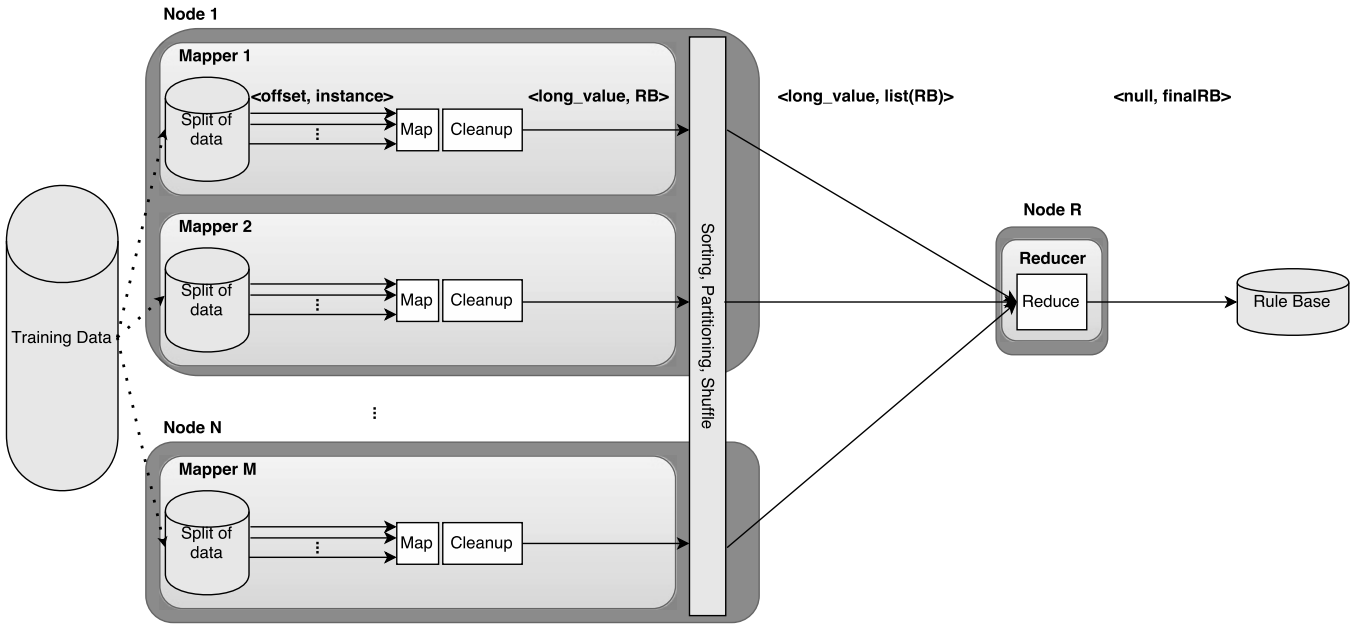


Fig. 2. Data flow of CHI^{BD}_{Local} algorithm.

reduction methods [24,25], clustering [26–28], Bayesian networks [29,30], rule induction algorithms [31], Extreme Learning Machines (ELMs) [32–34], and Support Vector Machines (SVMs) [35,36].

In general (and roughly speaking), there are two different approaches to adapt machine learning algorithms to MapReduce. Firstly, we have some methods that learn an independent model in each mapper to finally aggregate them forming an ensemble [35,37,14]. This methodology provides very good runtimes, but the accuracy might drop as we add nodes to the cluster. The reason is that each model is learned using a subset of the training set, and thus the reliability of the model will be dependent on the sampling and distribution of those subsets. The other approach is to concatenate multiple MapReduce jobs in order to run iterative learning processes [31]. In this case, the information needed for a single iteration can be processed in different nodes (running one or more MapReduce jobs) in the same manner as in the sequential version. This approach provides the same accuracy regardless of the number of nodes, but the overhead in each iteration might be substantial. Consequently, the scalability of this methodology is not clear when the number of iterations required by the algorithm increases.

Following these distributed models, several proposals have been presented in the literature. In [24] authors applied the concept of Minimal Consistent Subset (MCS) along with fuzzy boundaries to Hyper Surface Classification (HSC) in order to present a new parallel sampling method (PSHS). With a similar purpose but focusing on a different approach, Triguero et al. [25] presented some parallel versions of different prototype reduction algorithms and combined them to obtain a trade-off between accuracy and runtime. Regarding clustering techniques, a distributed version of the K-Means algorithm was introduced in [28], while Kim et al. proposed a density-based clustering algorithm for MapReduce in [27]. Focusing on the load balancing, a parallel version of the DBSCAN algorithm based on computation cost estimation was presented in [26]. In the field of Bayesian networks, Yue et al. proposed a distributed and incremental learning method for Bayesian networks extending a MDL-based scoring and search algorithm in [30]. An application of parallel Bayesian networks to financial factor relationships learning can be found in [29]. With respect to ELMs, in [32] authors proposed a distributed solution for kernel matrix computation for ELMs. Following a similar research line, Wang et al. introduced a parallel online sequential extreme learning machine (POS-ELM) [34]. In [33], a MapReduce-based framework for ELM training in large-scale classification and regression problems is provided.

Otherwise, SVMs have been one of the most commonly adapted machine learning techniques to MapReduce. In general, there are two popular approaches to distribute their execution, even though both can be seen as an ensemble of SVMs. The first one (and the most common one) consists of dividing the training set into multiple SVMs [38], building a hierarchy where SVMs of the new layers are learned using only the support vectors found in the SVMs of the previous ones. The process ends in the last layer where only one SVM is learned. This solution is limited by the fact that training data must be shared among the different nodes, notably increasing the distributing overhead.

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

The second approach consists of learning independent SVMs, aggregating (normally with the arithmetic mean) all the parameters learned in each one. In this case, the classification performance significantly drops as the number of nodes increases. Both SVMs models have been applied in different areas such as protein–protein interaction prediction [36] and spam filtering [35].

Regarding FRBCSs for Big Data, few MapReduce-based solutions have been published to date [37,14]. In [39], the authors present an overview of the current situation of fuzzy systems in Big Data scenarios. Nonetheless, these approaches have some scalability limitations in terms of classification accuracy due to the way in which rule weights are computed, as it was explained in Section 2.3. For this reason, in this paper we aim at obtaining exactly the same model as the one we would obtain with the original Chi et al. algorithm, regardless of the number of nodes used for the execution.

3. CHI-BD (CHI^{BD}_{Global}): designing a new MapReduce solution for Big Data classification problems

The main objective of this paper is to present a new distributed FRBCS for Big Data classification problems based on the Chi classifier [15] named CHI-BD (in this paper renamed as CHI^{BD}_{Global}). Next, we describe the motivation of this work (Section 3.1), the basic idea of the method (Section 3.2), the MapReduce implementation (Section 3.3), and a boosted version of CHI^{BD}_{Global} that speeds up the execution of the algorithm (Section 3.4).

3.1. Motivation

As we described in Section 2.3, the rule weights generated by CHI^{BD}_{Local} strongly depend on the number of mappers used for the execution of the algorithm. The reasons are the following ones:

- Each mapper computes the rule weights considering only a subset of the training set. As a result, the quality of these weights strongly depends on the distribution of the given subset and decreases as more computing nodes are added.
- The small sample size problem [39] notably affects the rule weights. In binary datasets, in the hypothetical case that the number of mappers is higher than the number of instances of a certain class, there will be mappers with no representation of that class. This implies that the rule weights generated in those mappers will be insignificant (all of them will be 1), since all the examples belong to the same class. Moreover, these rules will be kept in the final rule base, since they have the maximum possible weight.

For these reasons, we aim at designing a new computational solution that allows one to obtain exactly the same model regardless of the configuration of the cluster. In this manner, we can ensure that the degree of parallelism will not affect the classification performance of the algorithm. To this end, we propose to exploit the potential of MapReduce by taking advantage of the suitability of the Chi et al. algorithm for this framework.

3.2. Basic idea

As we described in Section 2.2, the original Chi algorithm generates a new fuzzy rule for each input example. This means that the rule generation process (without considering rule weights) does not depend on the entire training set. This fact makes Chi an exceptional candidate to be adapted to the MapReduce framework, even though the previous approach [14] did not consider it. Our aim is to fully exploit this fact in the proposed CHI^{BD}_{Global} method consisting of two different stages, where each one represents a single MapReduce job.

1. *Rules generation process*: in this step, all the fuzzy rules are created without computing their weights, obtaining a preliminary rule base. To this end, a new rule is generated for each instance in each mapper. In this manner, the output of each mapper are multiple $< antecedents, consequent >$ pairs (as many as the number of rules obtained). Finally, the reducers automatically group the rules provided by the mappers, obtaining a list of all the possible consequents ($list(consequent)$) for each rule ($antecedents$).
2. *Computation of rule weights*: the rule base obtained is shared by all the nodes, which allows us to compute the rule weights using all the instances. In order to do so, in each single mapper, the matching degree of all the rules

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

with the instances in the mapper are computed. Thereafter, all these matching degrees are used by the reducers to compute the final weight of each rule (automatically deciding the final consequent from the list of classes).

3.3. MapReduce implementation

In what follows, an in-depth description of all the MapReduce jobs involved in the two stages of CHI^{BD}_{Global} is presented. Algorithms 4–8 depict the pseudo-code of the corresponding MapReduce jobs, while Fig. 3 shows the data flow of both stages. The meaning of all the abbreviations shown in Fig. 3 are the following ones.

- *values*: values of the attributes of a given instance.
- *label*: class label of a given instance.
- *ants*: antecedents of a given rule.
- *class*: consequent class of a given rule.
- *ruleID*: ID of a given rule.
- *match*: accumulative matching degree of a given rule for a certain class.

3.3.1. Rule generation process

In this stage the initial rule base is generated without rule weights, and consequently there may be different candidates to be the consequent class of the rule. In order to do so, a new rule is constructed for each input instance in each mapper (Algorithm 4, line 1) and the antecedent and consequent parts of the rule generated are returned (Algorithm 4, line 2). Finally, the combiner and the reducer group the rules with the same antecedent part and store the different consequents (Algorithm 5). In this case, the code of both the combiner and the reducer is the same.

Algorithm 4 Mapper map() function for the rule generation process in CHI^{BD}_{Global}.

Procedure map (key, value)

Input: <*key*, *value*> pair, representing the values and the class of the instance, respectively.
Output: <*key*′, *value*′> pair, where *key*′ and *value*′ represent the antecedent part and the consequent (class) of the rule generated, respectively.
Begin
1: *ruleAnts* ← *generateRuleFromInstance* (*key*)
2: EMIT (*ruleAnts*, *value*)
End

Algorithm 5 Combiner/Reducer reduce() function for the rule generation process in CHI^{BD}_{Global}.

Procedure reduce (key, values)

Input: <*key*′, *values*> pair, where *key*′ represents the antecedents of a given rule and *values* is a list of the consequents of those rules with the same antecedent part as *key*′.
Output: <*key*″, *value*″> pair, where *key*″ is equal to *key*′ and *value*″ equals *values*.
Begin
1: EMIT (*key*′, *values*)
End

3.3.2. Computation of rule weights

After obtaining the antecedent and consequent parts of all the rules, we need to compute their weights. To do so, the matching degree of each example with every single rule must be computed. This process is concurrently performed using MapReduce, in such a way that each mapper computes the matching degree between the examples associated with its split and the entire rule base.

With this aim, each mapper loads the rule base generated in the previous stage. Then, each call to the map() function computes the matching degrees of the example with all the rules (Algorithm 6, line 2). Finally, once all the matching

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

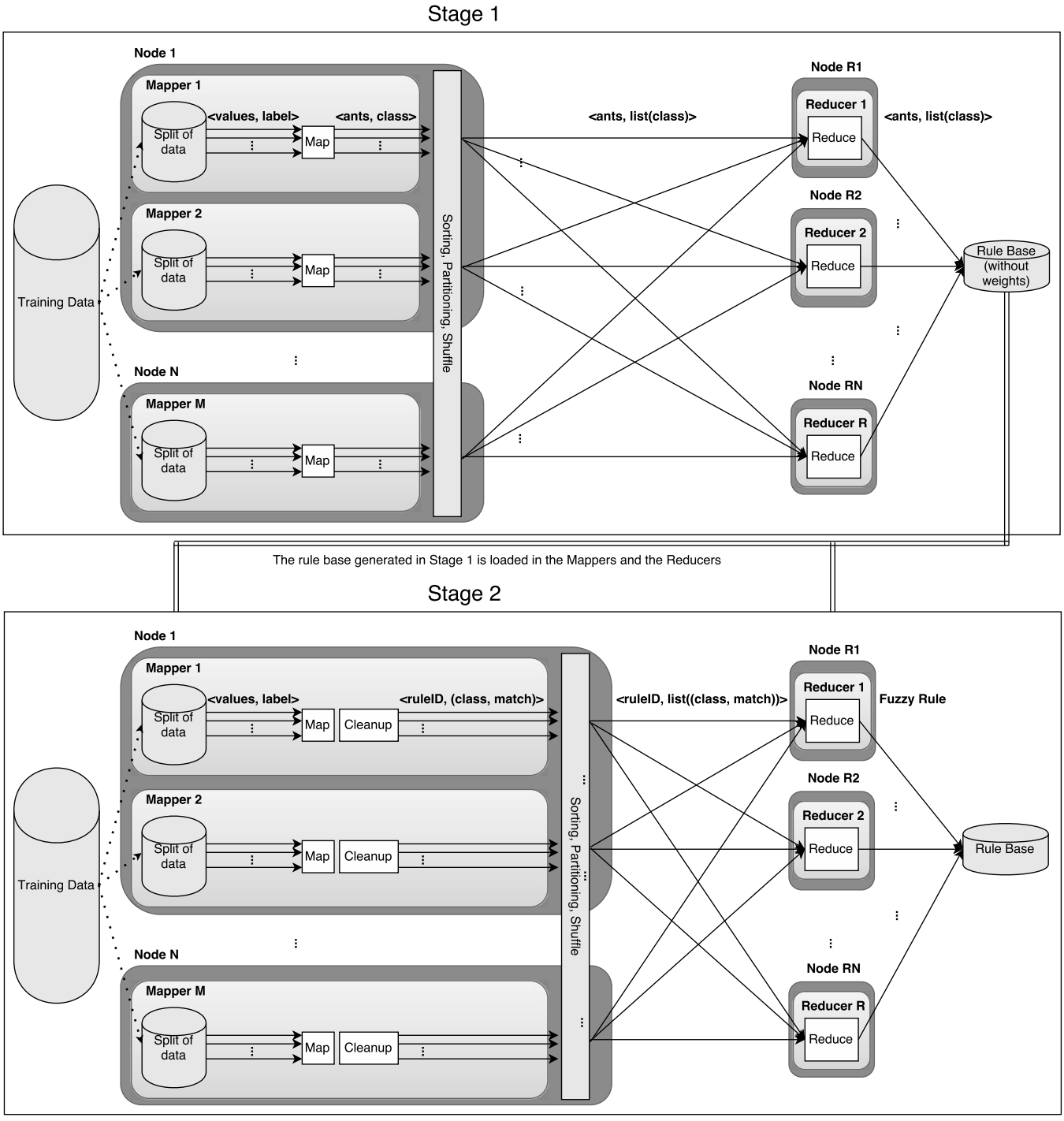


Fig. 3. Data flow of CHI^{BD}_{Global}.

degrees have been obtained, the cleanup() function returns a <key, value> pair for each rule (Algorithm 7, line 2) containing the following:

- key: an integer corresponding to the rule ID.
- value: an array composed of as many elements as the number of possible classes, where each element is the sum of the matching degrees of all the examples with the corresponding class. The role of this array can be understood by taking a look at Eq. (6). As we can observe, the weight of a given rule is computed by considering, on the one

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

hand, the matching degree of the examples belonging to the consequent and, on the other, the matching degree of the examples belonging to the rest of the classes.

We must remark that although the usage of the cleanup() function is not mandatory (the map() function can directly return the matching degree between the given instance and the whole rule base), the network traffic is notably reduced when it is used.

Finally, the matching degrees computed in all the mappers are summed and used to compute the rule weights in the reducer (Algorithm 8) applying the PCF (Eq. (2)) or PCF-CS (Eq. (6)). In this manner, the rule weights are obtained taking into account the whole training set. In order to select the final consequent in the rules where more than one candidate exists, that obtaining the highest rule weight is selected as in the original Chi et al. model (Algorithm 8, line 11). Rules with negative weight are removed from the rule base.

Since rule weights are computed considering the whole training set, the weights obtained will not vary when the number of nodes used for the execution changes. Moreover, all the possible consequent classes of a given rule are assigned to a single reducer, and thus the user can specify the number of reducers to be executed. More specifically, in our implementation the user sets the maximum number of rules to be processed in a single reducer and the application calculates how many reducers are required considering the rule base size. This feature notably speeds up the computation of rule weights.

Algorithm 6 Mapper map() function for the computation of rule weights in CHI^{BD}_{Global}.

Procedure map (key, value)

Input: <*key*, *value*> pair, representing the values and the class of the instance, respectively.
Begin
{Compute the matching degree of the example with all the rules generated in stage 1. The antecedents and consequents of these rules are stored in *RB* and *RBClasses* (used in the reducer), respectively}
1: **for** $i = 0$ **to** *RB.size* () $- 1$ **do**
2: *matchingDegrees* [*i*] [*value*] $\leftarrow$ *matchingDegrees* [*i*] [*value*] + *computeMatchingDegree* (*key, RB.get* (*i*))
3: **end for**
End

Algorithm 7 Mapper cleanup() function for the computation of rule weights in CHI^{BD}_{Global}.

Procedure cleanup ()

Output: <key', $value'$> pair, where key' is the ID of a given rule and $value'$ is a list containing the sum of the matching degrees for all the classes in the problem.
1: **for** $i = 0$ **to** *RB.size* () $- 1$ **do**
2: EMIT (*i*, *matchingDegrees* [*i*])
3: **end for**
End

3.4. *Boosting the execution of* CHI^{BD}_{Global}

In order to avoid repeated computations and speed up the execution of the CHI^{BD}_{Global} algorithm, we propose two pre-computation steps to be applied in the last stage of the method, that is, in the computation of rule weights:

1. *Pre-computation of membership degrees*. Since the rule weight computation requires the matching degree with all the examples, we observe that for the same example the membership degree to the same linguistic label is computed as many times as that label appears in all the fuzzy rules. This fact implies that the same computation is carried out repeatedly with the same result, and consequently a lot of computation time is wasted. In order to avoid this issue, we propose the usage of Look-Up-Tables to ensure that each membership degree is computed only once per example.

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

Algorithm 8 Reducer reduce() function for the computation of rule weights in CHI^{BD}_{Global}.

Procedure reduce (key, values)

Input: <*key*′, *values*> pair, where *key*′ is the ID of a given rule and *values* is a list of lists containing the sum of the matching degrees for all the classes in the problem.
Output: <*key*″, *value*″> pair, where *key*″ represents the antecedent and consequent part of the rule and *value*″ is the rule weight.
Begin
{Sum of the matching degrees for each class}
1: **while** *values.hasNext* () **do**
2: *currentMatching* ← *values.next* ()
3: **for** $i = 0$ **to** *NUM_CLASSES* − 1 **do**
4: *sumMatching* [*i*] ← *sumMatching* [*i*] + *currentMatching* [*i*]
5: **end for**
6: **end while**
{Compute the rule weight for each class and select the class with the highest rule weight}
7: *maxWeight* ← 0
8: *ruleClasses* ← *RBClasses.get* (*key*′) {*RBClasses* contains the list of consequents (classes) of each rule}
9: **for** $i = 0$ **to** *ruleClasses.size* () − 1 **do**
10: *weight* ← *computeRW* (*sumMatching*, *ruleClasses.get* (*i*))
11: **if** *weight* > *maxWeight* **then**
12: *maxWeight* ← *weight*
13: *maxClass* ← *ruleClasses.get* (*i*)
14: **end if**
15: **end for**
16: **if** *maxWeight* > 0 **then**
17: EMIT ((*RB.get* (*key*′), *maxClass*), *maxWeight*)
18: **end if**
End

2. *Pre-computation of partial matching degrees*. When computing the matching degree of a given example with all the rules, there might be multiple rules having several subsets of contiguous antecedents that are the same, resulting in a high number of repeated multiplications. For this reason, for each example, we propose to pre-compute the partial matching degrees of the subsets of antecedents that are considered to be frequent (using the Look-Up-Table where the pre-computed membership degrees are stored). To do so, in first place, we propose to find out these frequent subsets using an extra MapReduce stage (called *Frequent subsets search*). Then, the partial matching degrees of the obtained frequent subsets are computed before calculating the rule weights.

These extra steps allow one to substantially reduce the computing times, which are critical in Big Data environments.

As a result of these optimizations, the new boosted version of CHI^{BD}_{Global} comprises three different MapReduce stages (jobs):

1. *Rules generation process.* The preliminary rule base is learned as described in Section 3.3.1.
2. *Frequent subsets search.* A search of the frequent subsets of contiguous antecedents is carried out by the new MapReduce job.
3. *Computation of rule weights (including pre-computation of membership degrees and partial matching degrees).* The rule weights are computed making use of the preliminary rule base and the frequent subsets of antecedents computed in the previous phase to speed up the computation of the matching degrees.

As we can observe, the new MapReduce job is added between the first and second phases of the implementation described in Section 3.3 and the second phase (the third one from now on) is modified to include the pre-computation steps. Next, all these changes are described in detail.

The basic idea of the search of frequent subsets of contiguous antecedents is to count how many times the same subset appears in all the rules of the rule base. To this end, each rule is divided into a certain number of disjoint subsets of contiguous antecedents (*num_subsets*), as shown in Fig. 4. We will consider a subset to be frequent whenever its number of occurrences exceeds a threshold established by the user (*min_occurrences*). The influence of these parameters is explained in Section 4.2.

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

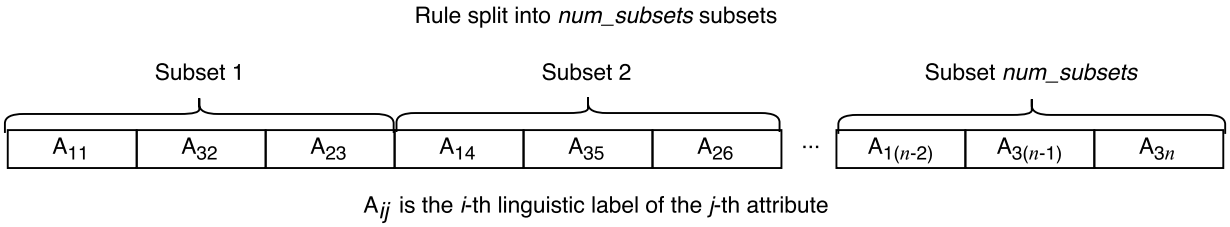


Fig. 4. Example of a rule split into *num_subsets* subsets.

Implementation details: Algorithms 9–11 show the pseudo-code of the new MapReduce job responsible for the search of frequent subsets. The map() function receives the preliminary rule base generated in the previous stage (Section 3.3.1) in order to divide the antecedent part of each of them into multiple disjoint subsets (Algorithm 9 and Fig. 4). Next, a new <key, value> pair is returned for each subset (Algorithm 9, line 4), where the key is a pair containing the antecedents and the number of the given subset, and the value is 1 (indicating a single occurrence). Finally, in both the combiner and the reducer the number of occurrences are summed (Algorithm 10, line 1 and Algorithm 11, line 3), and only those subsets of antecedents with a minimum number of occurrences (configurable by the user) are selected in the reducer (Algorithm 11, lines 5–6). In this manner, after the map phase we obtain the classic *word counter* for MapReduce, and thus we take advantage of all the power of this framework.

Algorithm 9 Mapper map() function for the frequent subsets search in CHI^{BD}_{Global}.

Procedure map (key, value)

Input: <*key*, *value*> pair, representing the antecedent part and the list of classes of a given rule, respectively.
Output: <*key*′, *value*′> pair, where *key*′ is a pair containing the antecedents and the number of a given subset and *value*′ is 1 (representing an occurrence).
Begin
1: *ants* ← *key.getAntecedents* ()
2: **for** *numSubset* = 1 **to** *num_subsets* **do**
3: *antsSubset* ← *getRuleSubset* (*ants, numSubset, LENGTH[numSubset]*) {*LENGTH* contains the length of each subset, computed at the beginning of the job}
4: EMIT ((*antsSubset*, *numSubset*), 1)
5: **end for**
End

Algorithm 10 Combiner reduce() function for the frequent subsets search in CHI^{BD}_{Global}.

Procedure reduce (key, values)

Input: <*key*′, *values*> pair, where *key*′ is a pair containing the antecedents and the number of the subset and *values* is a list of integers containing the number of occurrences of the given subset of antecedents (in the given subset).
Output: <*key*″, *value*″> pair, where *key*″ is equal to *key*′ and *value*″ is the sum of *values*.
Begin
1: EMIT (*key*′, *values.length*())
End

On the other hand, the last stage is also modified to take advantage of the frequent subsets and the pre-computations that are carried out in this phase. The map() function is composed of three steps (we should recall that the map function computes the matching degrees of the input example with all the rules):

1. All the membership degrees for each attribute of the example are computed (the pre-computed membership degrees are obtained).
2. All the partial matching degrees associated with the frequent subsets are computed (the pre-computed partial matching degrees are obtained).

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

Algorithm 11 Reducer reduce() function for the frequent subsets search in CHI^{BD}_{Global}.

Procedure reduce (key, values)

Input: <*key*′, *values*> pair (same as in the combiner).
Output: <*key*″, *value*″> pair (same as in the combiner).
Begin
1: *count* ← 0
2: **while** *values.hasNext* () **do**
3: *count* ← *count* + *values.next*()
4: **end while**
5: **if** *count* >= *min_occurrences* **then**
6: EMIT (*key*′, *count*)
7: **end if**
End

3. The matching degree of the example with each rule is computed. In order to do so, the pre-computed partial matching degrees are used when the rule has any frequent subset. Otherwise, the pre-computed membership degrees are considered.

Fig. 5 and Algorithm 12 show the data flow and the adaptation of the map() function (which substitutes the one described in Section 3.3.2) of the last stage. Fig. 6 depicts the data flow of the three different stages of CHI^{BD}_{Global}. The meaning of all the abbreviations shown in Fig. 6 are the following ones.

- *ants*: antecedents of a given rule.
- *class*: consequent class of a given rule.
- *subset*: subset of contiguous antecedents.
- *count*: number of occurrences of a given subset.

Algorithm 12 Mapper map() function for the computation of rule weights in the boosted version of CHI^{BD}_{Global}.

Procedure map (key, value)

Input: <*key*, *value*> pair, representing the values and the class of the instance, respectively.
Begin
{Step 1: compute the membership degree for each linguistic label}
1: *preComputedMemberships* ← *computeMembershipDegrees* (*currentInstance*)
{Step 2: compute the partial matching degree of the example with all the
frequent subsets of antecedents found in the previous stage (*freqSubsets*)}
2: **for** $i = 0$ **to** *freqSubsets.size* () − 1 **do**
3: *partialMatching* [*i*] ← *computePartialMatching* (*key, preComputedMemberships, freqSubsets* [*i*])
{Each element of *freqSubsets* is a pair containing the antecedents and the number of the subset, while
the elements of *partialMatching* are pairs containing the number of the subset and the partial matching degree}
4: **end for**
{Step 3: compute the matching degree of the example with all the rules generated in stage 1. The
antecedents and consequents of these rules are stored in *RB* and *RBClasses* (used in the reducer), respectively}
5: **for** $i = 0$ **to** *RB.size* () − 1 **do**
6: *matchingDegrees* [*i*] [*value*] ← *matchingDegrees* [*i*] [*value*] + *computeMatchingDegree* (*key, RB.get* (*i*), *partialMatching*)
7: **end for**
End

4. Experimental framework

In this section, we present the framework used to develop the experiments carried out in Section 5. Firstly, we describe the datasets selected for the experimental study (Section 4.1). Next, we show the parameters setup considered

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

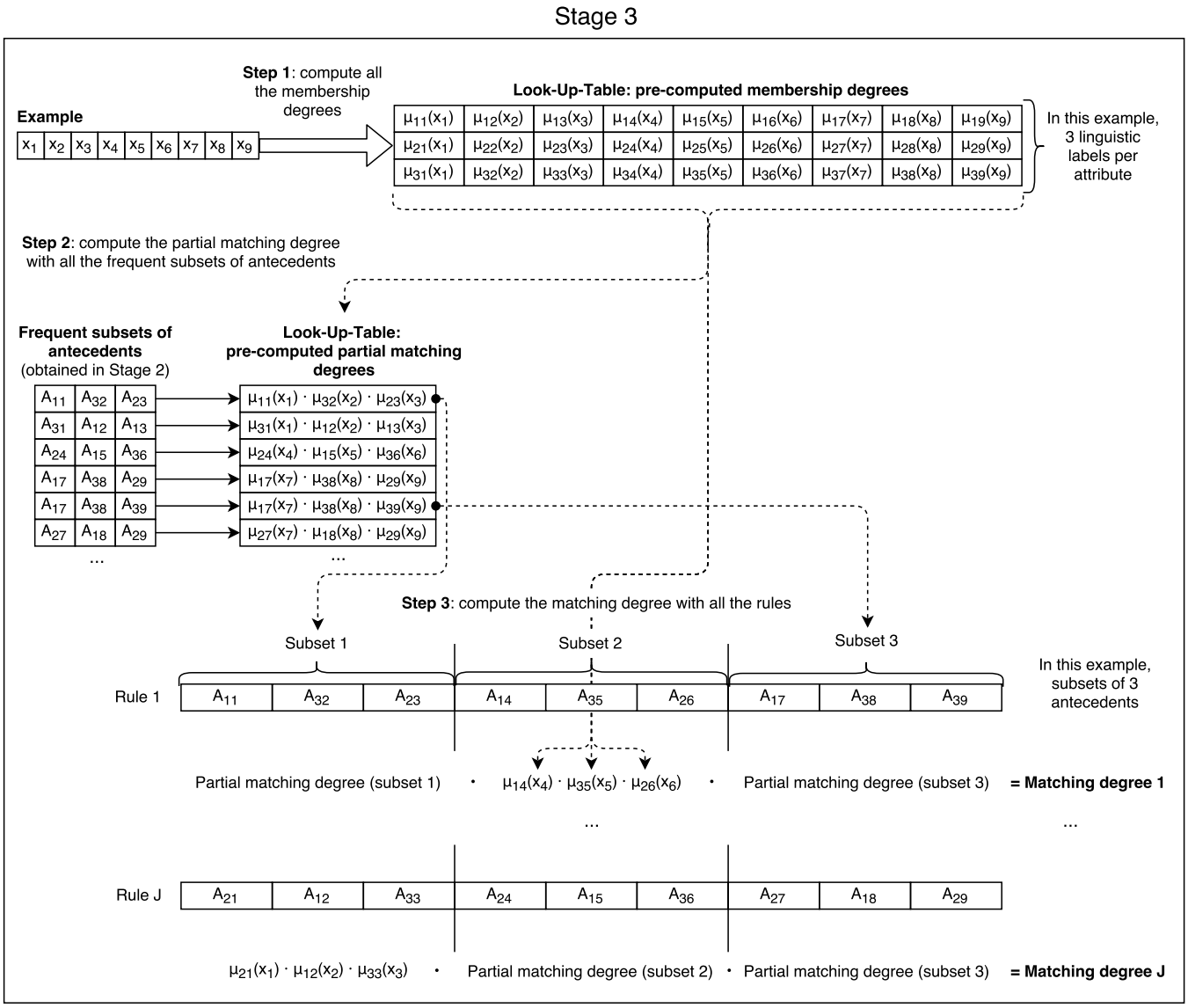


Fig. 5. Pre-computation steps in the learning process of CHI^{BD}_{Global}.

for each method (Section 4.2). Finally, we introduce the statistical tests used to compare the results obtained by the different methods, along with the performance and efficiency measures (Section 4.3).

The entire source code of CHI^{BD}_{Global} is publicly available at GitHub (https://github.com/melkano/chi-bd) under the GNU General Public License.

4.1. Datasets

In order to develop the experimental study, we have considered 8 different datasets from the UCI repository [17]. Aiming at obtaining more datasets and working with the same experimental framework as in [14], we have obtained 20 different binary classification problems by converting the original 8 datasets into multiple *One-vs-Rest* binary problems. To do so, in each original dataset we select a positive class and we consider the rest of classes as the negative one, in such a way that all the instances belong to the positive or the negative class. We consider this conversion in order to keep the size of the original dataset constant and thus, the resulting binary problems are still Big Data classification problems.

Most of these datasets are imbalanced classification problems, that is, the number of instances of the majority class is considerably higher than that of the minority one [23]. The main difficulty presented in this type of datasets is to

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

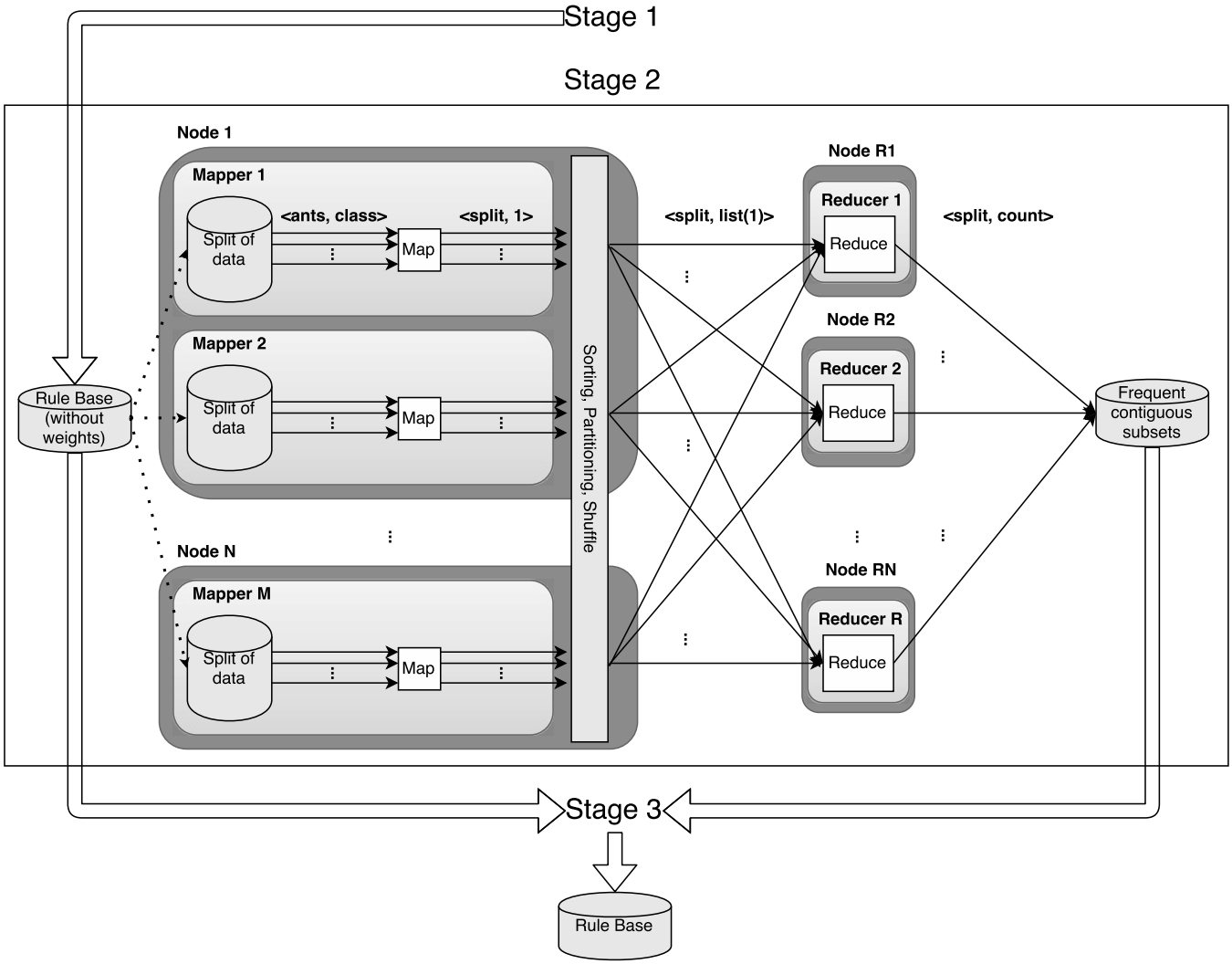


Fig. 6. Data flow of CHI^{BD}_{Global} (boosted version).

correctly identify those instances belonging to the minority class, since their misclassification cost is much higher than that of the instances belonging to the majority one. For this reason, we need to apply the cost-sensitive rule weight computation shown in Section 2.3 (Eq. (6)) in order to handle this type of problems, as we will explain in Section 4.2.

Table 1 shows the description of the datasets indicating the number of instances (#Instances), the number of instances of the majority and minority classes (P_{maj}:P_{min}), and the number of attributes (#Attributes). The abbreviations used for the different datasets are the followings: Cov (Covtype), Far (Fars), Kdd (KDDCup1999), and Pok (Poker). In the case of class labels, we use the following shortcuts: Fat (Fatal_Injury), Inc (Incapaciting_Injury), No (No_Injury), Nin (Nonincapaciting_Evident_Injury), nor (normal), and prb (probe). The resulting name is the abbreviation of the dataset plus the positive class shortcut.

All the experiments have been carried out using a *5-fold stratified cross-validation scheme*. In this model, we randomly split the dataset into five partitions of data, each one containing 20% of the examples, and we employ a combination of four of them (80%) to train the system and the remaining one to test it. Therefore, the result for each dataset is computed as the average of the five partitions.

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

Table 1
Description of the datasets.

Dataset	#Instances	$(P_{maj}:P_{min})$	#Attributes			
			Real	Integer	Nominal	Total
Census	142,521	(134,359 : 8,162)	1	12	28	41
Cov_1	581,012	(369,172 : 211,840)	10	0	44	54
Cov_2	581,012	(297,711 : 283,301)	10	0	44	54
Cov_3	581,012	(545,258 : 35,754)	10	0	44	54
Cov_7	581,012	(560,502 : 20,510)	10	0	44	54
Far_Fat	100,968	(58,852 : 42,116)	5	0	24	29
Far_Inc	100,968	(85,896 : 15,072)	5	0	24	29
Far_No	100,968	(80,961 : 20,007)	5	0	24	29
Far_Nin	100,968	(87,078 : 13,890)	5	0	24	29
Higgs	11,000,000	(5,829,123 : 5,170,877)	28	0	0	28
Kdd_dos	4,898,431	(3,883,370 : 1,015,061)	26	0	15	41
Kdd_nor	4,898,431	(3,925,650 : 972,781)	26	0	15	41
Kdd_prb	4,898,431	(4,857,329 : 41,102)	26	0	15	41
Kdd_r2l	4,898,431	(4,897,305 : 1,126)	26	0	15	41
Pok_0	1,025,009	(513,701 : 511,308)	0	10	0	10
Pok_1	1,025,009	(591,912 : 433,097)	0	10	0	10
Pok_2	1,025,009	(976,181 : 48,828)	0	10	0	10
Pok_3	1,025,009	(1,003,375 : 21,634)	0	10	0	10
Skin	245,057	(194,198 : 50,859)	0	3	0	3
Susy	5,000,000	(2,712,173 : 2,287,827)	18	0	0	18

Table 2
Parameters setup for each method.

Algorithm	*Parameters*
Chi	Number of linguistic labels per variable = 3 Inference = Winning Rule Rule weight = PCF-CS
CHI^{BD}_{Global}	Number of rule subsets = 4 Minimum occurrences for frequent subsets = 10 Maximum rules per reducer = 400,000

4.2. Parameters setup

Table 2 shows the configuration and parameters that we have considered for each method for all problems. All the parameters specified for the original Chi algorithm are also used in the rest of methods, since all of them are based on Chi.

In the case of CHI^{BD}_{Global}, the optimal number of rule subsets depends on the number of attributes and the distribution of the instances in each dataset. If the length of each subset is too small and the number of occurrences is not high enough, the proportion of pre-computed multiplications will decrease. However, if we take too large subsets, the likelihood of occurrence of the subsets of antecedents will be notably lower. Consequently, we need to select the number of subsets that offers the best overall runtime. In order to do so, we carried out some preliminary tests which reveal that 4 rule subsets provide a good trade-off for the datasets considered in this work. Similarly, if the minimum number of occurrences to consider a subset of antecedents to be a frequent subset is too low, the overhead of storing/accessing a large array could be greater than the benefit obtained from pre-computations. In relation to the maximum number of rules to be processed by each single reducer, we should consider the minimum number of rules that allows one to maximize the parallelization (number of reducers) without increasing the communication overhead. In this work, this parameter has been set based on our cluster configuration and some preliminary tests.

With respect to CHI^{BD}_{Local}, the learning process of this algorithm may be too time-consuming when tackling large datasets, since some important features of MapReduce are not fully exploited. For this reason, we have applied the methodology presented for CHI^{BD}_{Global} in the original CHI^{BD}_{Local} algorithm in order to speed up its execution without

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

Table 3
Confusion matrix for a binary problem.

	Positive prediction	Negative prediction
Positive class	True Positive (TP)	False Negative (FN)
Negative class	False Positive (FP)	True Negative (TN)

altering the rule base (and thus the results obtained). In our implementation, we take advantage of key-value pairs to allow the algorithm to run multiple reducers and we include the pre-computation of membership degrees to avoid repeated computations. Consequently, all the execution times associated with CHI^{BD}_{Local} correspond to our own implementation of this method.

Regarding the infrastructure used to carry out the experiments, all the parallel methods have been executed in an 8 nodes cluster connected via 1Gb/s Ethernet LAN network. Half of these nodes are composed of 2 Intel Xeon E5-2620 v3 processors at 2.4 GHz (3.2 GHz with Turbo Boost) with 12 virtual cores in each one (where 6 of them are physical). Three of the remaining nodes are equipped with 2 Intel Xeon E5-2620 v2 processors at 2.1 GHz with the same number of cores as the previous ones. The last node is the master node, composed of an Intel Xeon E5-2609 processor with 4 physical cores at 2.4 GHz. All slave nodes are equipped with 32 GB of RAM memory, while the master works with 8 GB of RAM memory. With respect to the storage specifications, all nodes use Hard Disk Drives featuring a read/write performance of 128 MB/s. The entire cluster runs under CentOS 6.5 and Apache Hadoop 2.6.0. This configuration features up to 42 concurrent YARN containers, where each container can be a mapper, a reducer, or the Application Master. However, we set a maximum of 32 concurrent mappers in order to have a power of two. In the case of the sequential Chi algorithm, the execution was performed in a single Intel Xeon E5-2620 v2 processor at 2.1 GHz.

4.3. Performance metrics, efficiency measures, and statistical tests

In order to evaluate the quality of the different methods, we test both the classification performance and the efficiency. Regarding the first aspect, since most of the datasets considered in this work are imbalanced binary problems, we should consider metrics that take into account the specific class distribution of each problem. With this purpose, we use the confusion matrix (Table 3) to obtain the number of correctly classified and misclassified examples for each class. From this matrix we obtain the following four metrics:

- True positive rate: percentage of correctly classified positive instances. $TP_{rate} = \frac{TP}{TP+FN}$.
- True negative rate: percentage of correctly classified negative instances. $TN_{rate} = \frac{TN}{TN+FP}$.
- False positive rate: percentage of misclassified negative instances. $FP_{rate} = \frac{FP}{FP+TN}$.
- False negative rate: percentage of misclassified positive instances. $FN_{rate} = \frac{FN}{FN+TP}$.

However, the previously mentioned metrics describe the classification accuracy for a certain class, but they do not combine the results of both classes. In order to consider the accuracy for both classes, we consider two commonly used metrics in this context [40], i.e., the Area Under the ROC Curve (AUC) [41] and the Geometric Mean (GM) [42], which are defined as:

$$AUC = \frac{1 + TP_{rate} + FP_{rate}}{2} \tag{7}$$

$$GM = \sqrt{TP_{rate} \cdot TN_{rate}} \tag{8}$$

With respect to the efficiency, we apply three well-known metrics used to evaluate distributed systems, i.e., speedup, sizeup, and scaleup [18,19]. We must point out that in our experiments one core is equivalent to one mapper.

- *Speedup*: the data size is kept constant and the number of cores is increased. An ideal distributed algorithm should feature a linear speedup, that is, a system with m cores must provide a speedup of m. However, in practice a linear speedup is difficult to obtain due to communication and synchronization overhead.

$$Speedup(m) = \frac{\text{runtime on 1 core}}{\text{runtime on } m \text{ cores}} \tag{9}$$

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

- *Sizeup*: the number of cores is kept constant and the data size is increased. Sizeup measures how much longer it will take to process a m-times larger dataset.

$$Sizeup(data, m) = \frac{\text{runtime for processing } m \cdot data}{\text{runtime for processing } data} \tag{10}$$

- *Scaleup*: the ability of a system to run a m-times greater job with m-times large system is measured, whose ideal value should be 1 (runtime of the baseline system).

$$Scaleup(data, m) = \frac{\text{runtime for processing } data \text{ on 1 core}}{\text{runtime for processing } m \cdot data \text{ on } m \text{ cores}} \tag{11}$$

We must remark that all the execution times of CHI^{BD}_{Global} shown in the experimental study correspond to the boosted version, i.e., the one including the pre-computation steps described in Section 3.4.

Aiming at giving statistical support to the analysis of the results, we use the Wilcoxon signed-ranks test [43] as a non-parametric statistical procedure to perform pairwise comparisons between two methods, as recommended in the specialized literature [44]. A complete description of this type of tests and software for their use can be found on the website available at http://sci2s.ugr.es/sicidm/.

5. Experimental study

In this section, we analyze the results obtained by our proposal developing an experimental study composed of the following steps:

1. We test the original Chi algorithm with the reduced versions of two Big Data classification problems, such as Higgs and Susy, and we compare it against CHI^{BD}_{Global} in terms of runtime (Section 5.1).
2. We show the benefits provided by the boosted version of CHI^{BD}_{Global} with respect to the non-boosted one (Section 5.2).
3. We analyze the classification performance and runtime of CHI^{BD}_{Global} (Section 5.3). We empirically show that the accuracy of CHI^{BD}_{Global} is the same as the one of the original Chi and that they are more accurate than the other Big Data alternative (CHI^{BD}_{Local}) as the number of mappers increases. The runtimes of both Big Data alternatives are also compared.
4. We study the efficiency of CHI^{BD}_{Global} in terms of speedup, sizeup, and scaleup [18,19] (Section 5.4).

5.1. The original Chi algorithm in Big Data

In order to study the ability of Chi to deal with Big Data classification problems, we have tested the original sequential algorithm considering the reduced versions of two large datasets, namely Higgs (11,000,000 instances and 28 attributes) and Susy (5,000,000 instances and 18 attributes). With this purpose, we have generated 4 reduced and stratified datasets for each of them. In this manner, we obtain the 1%, 5%, 10%, and 20% of the original datasets maintaining the same class distribution. Since both Chi and CHI^{BD}_{Global} obtain exactly the same model (and thus the same classification results), we only compare runtimes.

Table 4 shows the execution time of both the sequential Chi algorithm and CHI^{BD}_{Global} (using 32 mappers, the maximum we can run in parallel in our cluster). "ND" means that the execution was canceled after 18 days. We must stress that in the case of Susy, the overhead introduced by the MapReduce stages is higher than the processing time itself, and thus there is almost no difference among the different sizes. These results show that the original Chi algorithm is not able to address Big Data problems within a reasonable period of time. On the contrary, our approach is capable of dealing with large datasets featuring reasonable runtimes and providing the same classification results.

5.2. Pre-computation benefits

In order to highlight the importance of the implementation phase when working with Big Data problems, we show the benefits provided by the boosted version of CHI^{BD}_{Global} by performing a comparison against the non-boosted one.

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

Table 4
Runtimes (hh:mm:ss) for the original Chi algorithm and CHI^{BD}_{Global} (32 mappers).

Dataset	CHI	CHI^{BD}_{Global}
Higgs_1	41:07:53	00:01:18
Higgs_5	ND	00:03:08
Higgs_10	ND	00:06:59
Higgs_20	ND	00:18:52
Susy_1	00:33:32	00:00:56
Susy_5	03:41:34	00:00:59
Susy_10	09:54:33	00:00:57
Susy_20	24:44:21	00:00:58

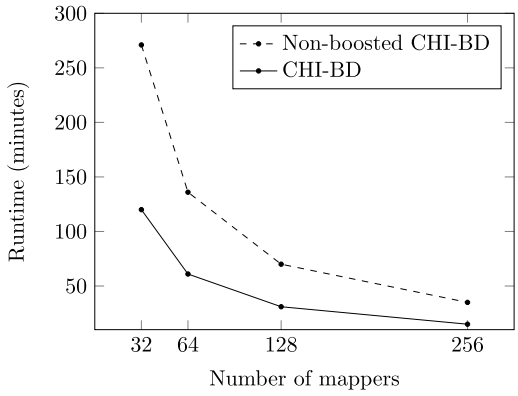


Fig. 7. Mappers runtime average in boosted and non-boosted CHI^{BD}_{Global} versions for Higgs.

As we showed in Section 3.4, in the boosted version another MapReduce job is executed in order to find the most frequent subsets of antecedents in the rule base. In this manner, the pre-computation is performed by computing the membership degrees with all linguistic labels and the partial matching degrees with these subsets, only once for each example (instead of computing it for all the rules). We must stress that the non-boosted version of CHI^{BD}_{Global} includes the pre-computation associated with the membership functions, and hence the difference between the two versions are the usage of pre-computed partial matching degrees, that is, the second MapReduce stage.

Aiming at comparing the boosted and non-boosted versions, we consider the largest and most complex dataset used in this experimental study, i.e., Higgs. The difference between the runtimes of the two versions can be visually appreciated in Fig. 7. As we can observe, we save half of the execution time when we apply the pre-computed partial matching degrees. This phase can avoid millions of multiplications when computing matching degrees.

5.3. *Performance of* CHI^{BD}_{Global}

In this section we test both the classification performance and the runtime of our approach and we compare it against the other existing Big Data adaptation of Chi in the literature, i.e., CHI^{BD}_{Local} [14].

5.3.1. *Classification performance*

Table 5 shows the GM and AUC obtained by CHI^{BD}_{Global} and CHI^{BD}_{Local} running 32, 64, 128, and 256 mappers for each of the 20 datasets. Since in the case of CHI^{BD}_{Global} the classification performance is not affected by the number of mappers, there is only one column corresponding to CHI^{BD}_{Global}. The best overall result for each dataset is shown in bold-face, while the best one among the different number of mappers is underlined. As one can observe, in the case of CHI^{BD}_{Local} the classification performance is clearly affected by the number of mappers, since adding more mappers is translated into a lower classification performance. When we consider the maximum number of mappers used in this study (256), the GM and AUC drop in average by 13% and 6% with respect to CHI^{BD}_{Global}, respectively. The reasons for this negative behavior are the fact that the rule weights are computed using portions of the original data and the small sample size problem, as explained in Section 2.3.

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

Table 5
GM and AUC obtained in testing by each method.

(a) GM

Dataset	CHI^{BD}_{Global}	CHI^{BD}_{Local}			
		32	64	128	256
Census	**.5231**	.4030	.4058	.4113	.4115
Cov_1	**.7531**	.7528	.7523	.7470	.7350
Cov_2	.7291	**.7296**	.7278	.7264	.7246
Cov_3	**.9565**	.9551	.9456	.9312	.9210
Cov_7	**.9281**	.9089	.8886	.8587	.8322
Cov AVG.	**.8417**	.8366	.8286	.8158	.8032
Far_Fat	**.5871**	**.5871**	**.5871**	.5870	.5870
Far_Inc	**.6919**	.5572	.5549	.5561	.5595
Far_No	**.8675**	.8336	.8338	.8272	.8274
Far_Nin	**.7139**	.5307	.5408	.5397	.5426
Far AVG.	**.7151**	.6271	.6292	.6275	.6291
Higgs	**.5847**	.5772	.5736	.5691	.5637
Kdd_dos	**.9991**	**.9991**	**.9991**	**.9991**	**.9991**
Kdd_nor	**.9992**	**.9992**	**.9992**	**.9992**	**.9992**
Kdd_prb	**.9924**	.9911	.9910	.9900	.9860
Kdd_r2l	**.9840**	.9251	.8769	.8094	.4232
Kdd AVG.	**.9937**	.9786	.9665	.9494	.8519
Pok_0	**.6336**	.6183	.6082	.5973	.5868
Pok_1	**.5848**	.5616	.5483	.5301	.5081
Pok_2	**.6703**	.3713	.2302	.1256	.1119
Pok_3	**.7387**	.2720	.1247	.0720	.0936
Pok AVG.	**.6569**	.4558	.3778	.3312	.3251
Skin	**.9597**	.9595	.9590	.9590	.9588
Susy	**.5524**	.5477	.5459	.5424	.5399
AVG.	**.7725**	.7040	.6846	.6689	.6456

(b) AUC

Dataset	CHI^{BD}_{Global}	CHI^{BD}_{Local}			
		32	64	128	256
Census	**.6220**	.5757	.5768	.5791	.5787
Cov_1	**.7532**	.7530	.7528	.7483	.7392
Cov_2	.7373	**.7379**	.7365	.7353	.7325
Cov_3	**.9572**	.9553	.9456	.9316	.9220
Cov_7	**.9285**	.9109	.8926	.8669	.8451
Cov AVG.	**.8441**	.8393	.8319	.8205	.8097
Far_Fat	**.6723**	.6722	**.6723**	.6721	**.6723**
Far_Inc	**.7123**	.6370	.6352	.6358	.6377
Far_No	**.8728**	.8438	.8440	.8383	.8386
Far_Nin	**.7283**	.6195	.6243	.6235	.6249
Far AVG.	**.7464**	.6931	.6939	.6924	.6933
Higgs	**.5848**	.5776	.5741	.5699	.5650
Kdd_dos	**.9991**	**.9991**	**.9991**	**.9991**	**.9991**
Kdd_nor	**.9992**	**.9992**	**.9992**	**.9992**	**.9992**
Kdd_prb	**.9925**	.9911	.9910	.9900	.9861
Kdd_r2l	**.9841**	.9279	.8844	.8280	.5924
Kdd AVG.	**.9937**	.9793	.9684	.9541	.8942
Pok_0	**.6360**	.6206	.6098	.5981	.5875
Pok_1	**.5859**	.5658	.5564	.5475	.5400
Pok_2	**.6709**	.5473	.5197	.5063	.5050
Pok_3	**.7388**	.5302	.5066	.5023	.5038
Pok AVG.	**.6579**	.5660	.5481	.5386	.5341
Skin	**.9605**	.9604	.9599	.9598	.9596
Susy	**.6242**	.6210	.6195	.6173	.6153
AVG.	**.7880**	.7523	.7450	.7374	.7222

Table 6
Wilcoxon tests to compare CHI^{BD}_{Global} and CHI^{BD}_{Local}.

CHI^{BD}_{Global} vs. CHI^{BD}_{Local}	GM		AUC	
	W/T/L	p-value	W/T/L	p-value
32 mappers	16/3/1	0.0002	17/2/1	0.0002
64 mappers	17/3/0	0.0001	17/3/0	0.0001
128 mappers	18/2/0	0.0001	18/2/0	0.0001
256 mappers	18/2/0	0.0001	17/3/0	0.0001

The previously mentioned factors make CHI^{BD}_{Local} not scalable in terms of classification performance. Moreover, this approach stores the entire split in memory, and thus there is a limit with respect to the vertical scalability (in terms of memory) as well.

In order to find significant differences between the classification performance of CHI^{BD}_{Global} and CHI^{BD}_{Local}, we have used the Wilcoxon signed-ranks test to compare both methods when running the different number of mappers considered. Table 6 depicts the results obtained in these comparisons, indicating the wins (W), ties (T), and losses (L) of CHI^{BD}_{Global} against CHI^{BD}_{Local} along with the computed p-value. As we can observe, our new approach statistically outperforms the CHI^{BD}_{Local} algorithm with a confidence level of 99% (p-value < 0.01) in all cases.

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

Table 7
Average runtime (hh:mm:ss) for CHI^{BD}_{Global} and CHI^{BD}_{Local}.

Dataset	Total runtime avg.		Mappers runtime avg.							
	CHI^{BD}_{Global}	CHI^{BD}_{Local}	CHI^{BD}_{Global}				CHI^{BD}_{Local}			
	32	32	32	64	128	256	32	64	128	256
Census	00:01:36	00:00:26	00:00:14	00:00:07	00:00:04	00:00:03	00:00:00	00:00:00	00:00:00	00:00:00
Cov_1	00:01:16	00:01:08	00:00:08	00:00:04	00:00:02	00:00:01	00:00:33	00:00:08	00:00:02	00:00:00
Cov_2	00:01:15	00:01:10	00:00:08	00:00:04	00:00:02	00:00:01	00:00:33	00:00:08	00:00:02	00:00:00
Cov_3	00:01:14	00:01:09	00:00:08	00:00:04	00:00:02	00:00:01	00:00:33	00:00:08	00:00:02	00:00:00
Cov_7	00:01:15	00:01:10	00:00:08	00:00:04	00:00:02	00:00:01	00:00:33	00:00:08	00:00:02	00:00:00
Cov AVG.	00:01:15	00:01:09	00:00:08	00:00:04	00:00:02	00:00:01	00:00:33	00:00:08	00:00:02	00:00:00
Far_Fat	00:01:21	00:00:24	00:00:06	00:00:04	00:00:02	00:00:01	00:00:00	00:00:00	00:00:00	00:00:00
Far_Inc	00:01:20	00:00:24	00:00:06	00:00:04	00:00:02	00:00:01	00:00:00	00:00:00	00:00:00	00:00:00
Far_No	00:01:22	00:00:23	00:00:06	00:00:04	00:00:02	00:00:01	00:00:00	00:00:00	00:00:00	00:00:00
Far_Nin	00:01:21	00:00:24	00:00:06	00:00:04	00:00:02	00:00:01	00:00:00	00:00:00	00:00:00	00:00:00
Far AVG.	00:01:21	00:00:24	00:00:06	00:00:04	00:00:02	00:00:01	00:00:00	00:00:00	00:00:00	00:00:00
Kdd_dos	00:01:18	01:12:42	00:00:21	00:00:10	00:00:05	00:00:03	00:58:35	00:10:55	00:02:21	00:00:29
Kdd_nor	00:01:16	01:11:57	00:00:21	00:00:10	00:00:05	00:00:03	00:56:50	00:10:55	00:02:21	00:00:29
Kdd_prb	00:01:17	01:13:22	00:00:21	00:00:10	00:00:05	00:00:03	00:58:55	00:10:49	00:02:21	00:00:29
Kdd_r2l	00:01:15	01:13:32	00:00:21	00:00:10	00:00:05	00:00:03	00:58:56	00:10:49	00:02:21	00:00:29
Kdd AVG.	00:01:16	01:12:53	00:00:21	00:00:10	00:00:05	00:00:03	00:58:19	00:10:52	00:02:21	00:00:29
Pok_0	00:01:47	00:00:58	00:00:27	00:00:14	00:00:07	00:00:03	00:00:25	00:00:06	00:00:01	00:00:00
Pok_1	00:01:50	00:00:59	00:00:28	00:00:14	00:00:07	00:00:04	00:00:25	00:00:06	00:00:01	00:00:00
Pok_2	00:01:49	00:00:57	00:00:27	00:00:14	00:00:07	00:00:04	00:00:26	00:00:06	00:00:01	00:00:00
Pok_3	00:01:46	00:00:59	00:00:27	00:00:13	00:00:07	00:00:03	00:00:25	00:00:06	00:00:01	00:00:00
Pok AVG.	00:01:18	00:00:58	00:00:27	00:00:14	00:00:07	00:00:03	00:00:25	00:00:06	00:00:01	00:00:00
Skin	00:00:53	00:00:22	00:00:00	00:00:00	00:00:00	00:00:00	00:00:01	00:00:00	00:00:00	00:00:00
Susy	00:01:43	00:33:39	00:00:37	00:00:18	00:00:09	00:00:05	00:26:49	00:05:46	00:01:14	00:00:16
AVG.	**00:01:24**	**00:17:41**	**00:00:15**	**00:00:08**	**00:00:04**	**00:00:02**	**00:13:53**	**00:02:38**	**00:00:34**	**00:00:06**
Higgs	02:40:58	04:11:55	01:59:45	01:00:54	00:31:07	00:15:24	03:43:58	00:48:11	00:10:15	00:01:52

5.3.2. *Runtime and complexity*

Besides classification performance, Big Data problems require algorithms that can be executed within a reasonable period of time. Accordingly, Table 7 shows the runtimes of both methods considering the different number of mappers. On the one hand, we provide the total execution time when using 32 mappers to run the entire MapReduce application (the maximum that can be executed in parallel in our cluster). On the other side, for greater numbers of mappers, we show the sum of the average runtime of the mappers corresponding to the three stages. The reason for not including the total time in these cases is that some mappers would not be executed in parallel, since our cluster supports 32 concurrent mappers at most, and thus the total time would not be reliable. The runtime of the reducers is not included either, since it is negligible compared with that of the mappers.

Looking at the description of the datasets in Table 1, we can observe that there is a huge difference between Higgs and the rest of datasets with respect to size and complexity. For this reason, we have not included the runtimes corresponding to Higgs in the average computation, and hence we show it separately in Table 7. As we can observe, CHI^{BD}_{Global} runs considerably faster than CHI^{BD}_{Local}, except for those datasets that are not large enough and the overhead of running three MapReduce jobs becomes higher than the processing time (in this case when the processing time is lower than 1 minute, approximately). The existing runtime difference between the two methods decreases as more mappers are added, due to the fact that in CHI^{BD}_{Global}, contrary to CHI^{BD}_{Local}, all the mappers work with the entire rule base. That is, the same number of computations are done in CHI^{BD}_{Global} regardless of the number of mappers, whereas in CHI^{BD}_{Local} they are reduced. Consequently, the speedup is lower in CHI^{BD}_{Global}, as we will explain later in the analysis

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

Table 8
Average number of rules for CHI^{BD}_{Global} and CHI^{BD}_{Local}.

Dataset	CHI^{BD}_{Global}	CHI^{BD}_{Local}			
		32	64	128	256
Census	63,598	64,137	64,145	64,151	64,152
Cov_1	7,940	8,275	8,398	8,501	8,598
Cov_2	8,108	8,372	8,446	8,541	8,610
Cov_3	8,249	8,438	8,468	8,519	8,570
Cov_7	7,917	8,181	8,297	8,413	8,526
Cov AVG.	8,053	8,317	8,402	8,494	8,576
Far_Fat	49,707	49,707	49,707	49,707	49,707
Far_Inc	49,130	49,692	49,701	49,706	49,708
Far_No	49,584	49,700	49,702	49,704	49,705
Far_Nin	48,984	49,686	49,695	49,703	49,705
Far AVG.	49,351	49,696	49,701	49,705	49,706
Kdd_dos	5,747	5,753	5,753	5,753	5,753
Kdd_nor	5,734	5,755	5,757	5,758	5,759
Kdd_prb	5,701	5,750	5,753	5,753	5,756
Kdd_r2l	5,686	5,744	5,750	5,752	5,753
Kdd AVG.	5,717	5,751	5,753	5,754	5,755
Pok_0	54,523	56,023	56,097	56,121	56,133
Pok_1	54,254	55,986	56,074	56,109	56,132
Pok_2	46,618	55,408	55,733	55,926	56,035
Pok_3	44,632	55,480	55,781	55,942	56,037
Pok AVG.	50,007	55,724	55,921	56,024	56,084
Skin	23	23	23	23	24
Susy	9,505	9,675	9,724	9,793	9,874
AVG.	**27,665**	**29,041**	**29,105**	**29,151**	**29,186**
Higgs	666,068	762,443	781,843	797,712	809,592

of the efficiency. However, when the number of mappers is high enough for CHI^{BD}_{Local} to run faster than CHI^{BD}_{Global}, the classification performance is no longer admissible.

When we work with FRBCSs, the number of rules generated is usually a good indicator of the complexity of the problem. Generally, a FRBCS will be able to solve simple classification problems using a low number of rules (with respect to the number of instances of the problem). Table 8 shows the number of rules obtained with each method when running different number of mappers. Since CHI^{BD}_{Global} provides the same model regardless of the number of mappers used for its execution, there is only one column for this method. In the case of CHI^{BD}_{Local}, however, the number of rules obtained grows as more mappers are added. The reason for this behavior is that reducing the number of instances in each mapper implies that less rules are obtained with negative rule weight, since there is a lower probability of finding counter-examples. Although CHI^{BD}_{Global} might suffer from a memory bottleneck caused by the fact that the entire rule base must be loaded in the main memory, we must remark that the 666,068 rules generated in Higgs occupy about 200 MB (using a byte to represent an antecedent and Java's primitive arrays to store rules), which represents 2.5% of the available main memory on a typical 8 GB RAM hardware. In future work, our aim is to tackle this bottleneck by applying feature selection approaches to reduce the size of the rule base before creating it and also to apply rule selection strategies to reduce the size of the final rule base. In this work we preferred to maintain the original Chi et al. model in order to show the differences between global and local models. As we will see in Section 5.4, the rule base size will play an important role in the scaleup of the model.

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

Table 9
Runtimes (hh:mm:ss) of CHI^{BD}_{Global} used for efficiency measures.

Data size	4 mappers	8 mappers	16 mappers	32 mappers
10%	00:01:19	00:00:42	00:00:20	00:00:11
20%	00:03:58	00:02:09	00:01:08	00:00:32
40%	00:12:21	00:06:00	00:03:09	00:01:38
80%	00:34:37	00:17:38	00:08:56	00:04:32

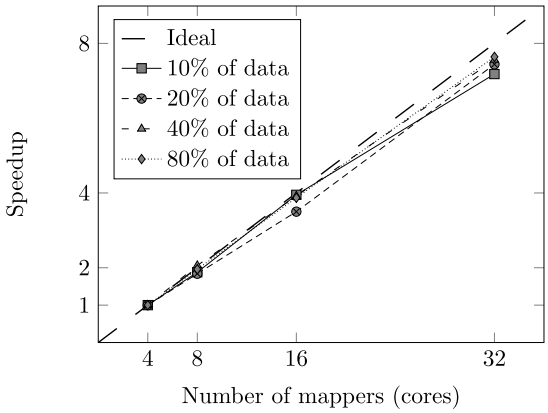


Fig. 8. Speedup performance of CHI^{BD}_{Global}.

5.4. Efficiency

Finally, we study the efficiency of CHI^{BD}_{Global} in terms of speedup, sizeup, and scaleup [18,19] by averaging the runtimes obtained in all datasets. In order to do so, we take 4 mappers and 10% of each dataset as the baseline case ($m = 1$) and we gradually double both the number of mappers and the data size (maintaining the original class distribution), until 32 mappers and 80% of data. In this manner, for each number of mappers (4, 8, 16, and 32) we run the model using 10%, 20%, 40%, and 80% of data. With the runtimes obtained in these executions we build the matrix shown in Table 9, which is used to compute the speedup, sizeup, and scaleup.

Figs. 8, 9, and 10 show the results obtained by the speedup, sizeup, and scaleup efficiency measures. In all these figures, the ideal case is indicated with a long dashed line. Next, we analyze each aspect separately.

- *Speedup*: as we can observe in Fig. 8, the speedup of CHI^{BD}_{Global} is almost the ideal one, obtaining around m times faster model when using m times more mappers.
- *Sizeup*: Fig. 9 depicts that the sizeup increases too quickly. The reason is that the execution time of the original Chi is clearly affected by complex problems. Since the computation of rule weights requires to calculate the matching degree of each rule with all the examples, the learning process might have order of $O(P \cdot numRules)$ time complexity, P and $numRules$ being the number of training examples and rules learned, respectively. As a consequence, the impact of the data size is not linear (contrary to the effect caused by the number of mappers, as shown in Fig. 8.
- *Scaleup*: the sizeup cannot reach the ideal value due to the inherent working procedure of Chi, and consequently scaleup is equally affected, as shown in Fig. 10. Nevertheless, we can observe that there is a huge difference between the decrease obtained between 10% and 20% and the rest. This is because the likelihood of having duplicated rules increases as the number of instances becomes higher (if the number of attributes remains constant, as in this case), reducing the time complexity as P becomes predominant over $numRules$. Consequently, we can expect that the scaleup of CHI^{BD}_{Global} will remain stable in the following scenarios:
  - The proportion of instances with respect to attributes is large enough to avoid an explosion of possible combinations of antecedents.
  - Input values show sufficient homogeneity to obtain a low proportion of rules with respect to the number of instances.

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

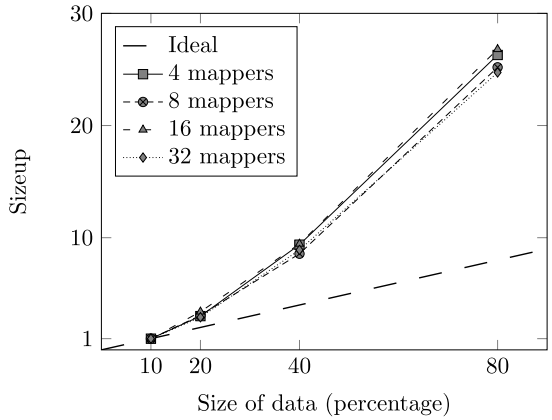


Fig. 9. Sizeup performance of CHI^{BD}_{Global}.

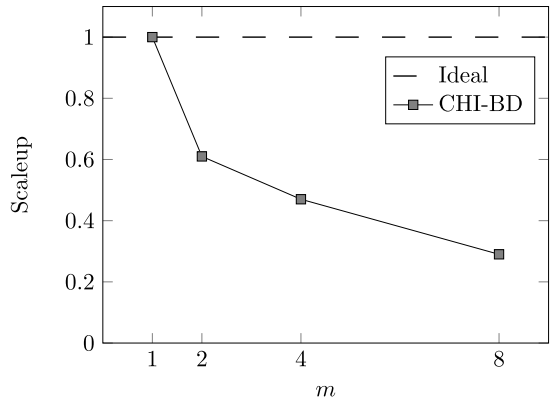


Fig. 10. Scaleup performance of CHI^{BD}_{Global}.

6. Concluding remarks

In this work we have presented a new distributed FRBCS for Big Data classification problems called CHI-BD (in this paper renamed as CHI^{BD}_{Global}. This method is based on the well-known Chi et al. algorithm and makes use of the most popular Big Data framework, i.e., Apache Hadoop.

As we have shown, the learning process of the original Chi et al. method takes more than 18 days in large classification problems, and thus the need for a new distributed approach arises. In this paper, we have introduced a new MapReduce solution that provides the same classification performance regardless of the number of mappers used for its execution. The fundamental feature that makes it possible is that Chi generates a new rule for each input example, allowing one to exploit the full potential of MapReduce. In this manner, we divide the learning process into two different stages (three in the boosted version) in order to distribute both the rule generation process and the computation of rule weights, obtaining exactly the same model as that provided by the original Chi algorithm. The experimental results show that CHI^{BD}_{Global} outperforms CHI^{BD}_{Local} method in terms of runtime and classification performance when dealing with Big Data problems.

In the future, the problem with the sizeup should be addressed. Since the original Chi algorithm computes the rule weights by calculating the matching degree of each rule with all the examples, CHI^{BD}_{Global} needs to deal with $O(P \cdot numRules)$ time complexity. This implies that an increase in the data size does not have a linear effect on the execution time, limiting the scaleup of the algorithm. In order to solve this problem, different approaches should be taken into account. Among them, a previous feature selection process (scalable for Big Data problems, as well) might be an interesting solution, increasing the likelihood of duplicated rules and reducing the rule base size. Another option could be to select a threshold that removes the rules generated by few examples.

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

The source code of CHI^{BD}_{Global} has been published at GitHub (https://github.com/melkano/chi-bd) under the GNU General Public License. In this manner, different researchers and practitioners can use, develop and improve the algorithm.

Acknowledgements

This work has been supported by the Spanish Ministry of Science and Technology under the projects TIN-2013-40765-P and TIN2016-77356-P.

References

[1] J. Gantz, D. Reinsel, The digital universe in 2020: big data, bigger digital shadows, and biggest growth in the far East, http://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf.
[2] L. Arthur, What is big data? Forbes, http://www.forbes.com/sites/lisaarthur/2013/08/15/what-is-big-data/.
[3] A. Gandomi, M. Haider, Beyond the hype: big data concepts, methods, and analytics, Int. J. Inf. Manag. 35 (2) (2015) 137–144.
[4] S. Nativi, P. Mazzetti, M. Santoro, F. Papeschi, M. Craglia, O. Ochiai, Big data challenges in building the global Earth observation system of systems, Environ. Model. Softw. 68 (0) (2015) 1–26.
[5] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
[6] B. Wilder, Cloud Architecture Patterns, O'Reilly Media, 2012.
[7] H. Ishibuchi, T. Nakashima, M. Nii, Classification and Modeling with Inguistic Information Granules: Advanced Approaches to Linguistic Data Mi Ning, Springer-Verlag, 2004.
[8] S. Ho, C. Hsieh, H. Chen, H. Huang, Interpretable gene expression classifier with an accurate and compact fuzzy rule base for microarray data analysis, Biosystems 85 (3) (2006) 165–176.
[9] P. Kumar, S. Vijay, D. Devaraj, A Hybrid Colony Fuzzy System for Analyzing Diabetes Microarray Data, 2013, pp. 104–111.
[10] J. Sanz, M. Galar, A. Jurio, A. Brugos, M. Pagola, H. Bustince, Medical diagnosis of cardiovascular diseases using an interval-valued fuzzy rule-based classification system, Appl. Soft Comput. J. 20 (2013) 103–111, http://dx.doi.org/10.1016/j.asoc.2013.11.009.
[11] C. Tsang, S. Kwong, H. Wang, Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection, Pattern Recognit. 40 (9) (2007) 2373–2391.
[12] J. Sanz, D. Bernardo, F. Herrera, H. Bustince, H. Hagras, A compact evolutionary interval-valued fuzzy rule-based classification system for the modeling and prediction of real-world financial applications with imbalanced data, IEEE Trans. Fuzzy Syst. 23 (4) (2014) 973–990, http://dx.doi.org/10.1109/TFUZZ.2014.2336263.
[13] T. Nakashima, G. Schaefer, Y. Yokota, A weighted fuzzy classifier and its application to image processing tasks, Fuzzy Sets Syst. 158 (3) (2007) 284–294.
[14] V. López, S. del Río, M. Benítez, F. Herrera, Cost-sensitive linguistic fuzzy rule based classification systems under the mapreduce framework for imbalanced big data, Fuzzy Sets Syst. 258 (0) (2015) 5–38.
[15] Z. Chi, H. Yan, T. Pham, Fuzzy Algorithms with Applications to Image Processing and Pattern Recognition, World Scientific, 1996.
[16] H. Ishibuchi, T. Yamamoto, Rule weight specification in fuzzy rule-based classification systems, IEEE Trans. Fuzzy Syst. 13 (4) (2005) 428–435.
[17] M. Lichman, UCI machine learning repository, http://archive.ics.uci.edu/ml, 2013.
[18] D. DeWitt, J. Gray, Parallel database systems: the future of high performance database systems, Commun. ACM 35 (6) (1992) 85–98.
[19] P. Jogalekar, M. Woodside, Evaluating the scalability of distributed systems, IEEE Trans. Parallel Distrib. Syst. 11 (6) (2000) 589–603.
[20] S. Ghemawat, H. Gobioff, S. Leung, The Google file system, in: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, SOSP '03, ACM, 2003, pp. 29–43.
[21] D. Paternain, H. Bustince, M. Pagola, P. Sussner, A. Kolesárová, R. Mesiar, Capacities and overlap indexes with an application in fuzzy rule-based classification systems, Fuzzy Sets Syst. 305 (2016) 70–94.
[22] O. Cordón, M. del Jesus, F. Herrera, A proposal on reasoning methods in fuzzy rule-based classification systems, Int. J. Approx. Reason. 20 (1) (1999) 21–45.
[23] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches, IEEE Trans. Syst. Man Cybern. 42 (4) (2011) 463–484.
[24] Q. He, H. Wang, F. Zhuang, T. Shang, Z. Shi, Parallel sampling from big data with uncertainty distribution, Fuzzy Sets Syst. 258 (2015) 117–133.
[25] I. Triguero, D. Peralta, J. Bacardit, S. García, F. Herrera, MRPR: a mapreduce solution for prototype reduction in big data classification, Neurocomputing 150 (PA) (2015) 331–345.
[26] Y. He, H. Tan, W. Luo, S. Feng, J. Fan, Mr-dbscan: a scalable mapreduce-based dbscan algorithm for heavily skewed data, Front. Comput. Sci. 8 (1) (2014) 83–99.
[27] Y. Kim, K. Shim, M.-S. Kim, J.S. Lee, Dbcure-mr: an efficient density-based clustering algorithm for large data using mapreduce, Inf. Sci. 42 (2014) 15–35.
[28] P. Zhou, J. Lei, W. Ye, Large-scale data sets clustering based on mapreduce and hadoop, J. Comput. Inf. Syst. 7 (16) (2011) 5956–5963.
[29] W. Chen, T. Wang, D. Yang, K. Lei, Y. Liu, Massively parallel learning of Bayesian networks with mapreduce for factor relationship analysis, in: Proceedings of the International Joint Conference on Neural Networks, 2013, pp. 1–5.

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

[30] K. Yue, Q. Fang, X. Wang, J. Li, W. Liu, A parallel and incremental approach for data-intensive learning of Bayesian networks, IEEE Trans. Cybern. 45 (12) (2015) 2890–2904, http://dx.doi.org/10.1109/TCYB.2015.2388791.
[31] V. Kolias, C. Kolias, I. Anagnostopoulos, E. Kayafas, Rulemr: classification rule discovery with mapreduce, in: Proceedings – 2014 IEEE International Conference on Big Data, 2015, pp. 20–28.
[32] X. Bi, X. Zhao, G. Wang, P. Zhang, C. Wang, Distributed extreme learning machine with kernels based on mapreduce, Neurocomputing 149 (PA) (2015) 456–463.
[33] J. Chen, H. Chen, X. Wan, G. Zheng, MR-ELM: a MapReduce-based framework for large-scale ELM training in big data era, Neural Comput. Appl. 27 (1) (2016) 101–110.
[34] B. Wang, S. Huang, J. Qiu, Y. Liu, G. Wang, Parallel online sequential extreme learning machine based on mapreduce, Neurocomputing 149 (3) (2015) 224–232.
[35] G. Caruana, M. Li, M. Qi, A mapreduce based parallel SVM for large scale spam filtering, in: 2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), vol. 4, 2011, pp. 2659–2662.
[36] Z.-H. You, J.-Z. Yu, L. Zhu, S. Li, Z.-K. Wen, A mapreduce based parallel SVM for large-scale predicting protein–protein interactions, Neurocomputing 145 (2014) 37–43.
[37] S. del Río, V. López, J.M. Benítez, F. Herrera, A mapreduce approach to address big data classification problems based on the fusion of linguistic fuzzy rules, Int. J. Comput. Intell. Syst. 8 (3) (2015) 422–437.
[38] H. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, V. Vapnik, Parallel support vector machines: the cascade SVM, Adv. Neural Inf. Process. Syst. (2005) 33–41.
[39] A. Fernandez, C. Carmona, M. del Jesus, F. Herrera, A view on fuzzy systems for big data: progress and opportunities, Int. J. Comput. Intell. Syst. 9 (1) (2016) 69–80.
[40] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes, Pattern Recognit. 44 (8) (2011) 1761–1776.
[41] J. Huang, C. Ling, Using auc and accuracy in evaluating learning algorithms, IEEE Trans. Knowl. Data Eng. 17 (3) (2005) 299–310.
[42] R. Barandela, J. Sánchez, V. García, E. Rangel, Strategies for learning in class imbalance problems, Pattern Recognit. 36 (3) (2003) 849–851.
[43] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics 1 (6) (1945) 80–83.
[44] J. Demšar, Statistical comparison of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.

Please cite this article in press as: M. Elkano et al., CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.07.003

4. CHI-PG: algoritmo para la generación rápida de prototipos en problemas de clasificación Big Data – *CHI-PG: A fast prototype generation algorithm for Big Data classification problems*

El trabajo asociado a esta parte es:

- M. Elkano, M. Galar, J. Sanz, H. Bustince "CHI-PG: A fast prototype generation algorithm for Big Data classication problems", Aceptado con revisiones menores en *Neurocomputing*, 2017.
    - Estado: Aceptado con revisiones menores.
    - Índice de Impacto (JCR 2016): 3,317.
    - Área de Conocimiento: Computer Science, Artificial Intelligence. Ranking 24/133 (Q1).

Elsevier Editorial System(tm) for Neurocomputing

Manuscript Draft

Manuscript Number: NEUCOM-D-17-02059R1

Title: CHI-PG: A fast prototype generation algorithm for Big Data classification problems

Article Type: Full Length Article (LS)

Keywords: Prototype Reduction; Prototype Generation; Big Data; MapReduce; Fuzzy Rule-Based Classification Systems

Corresponding Author: Mr. Mikel Elkano,

Corresponding Author's Institution: Universidad Pública de Navarra

First Author: Mikel Elkano

Order of Authors: Mikel Elkano; Mikel Galar; Jose Sanz; Humberto Bustince

Abstract: The growing amount of available data has become a serious challenge to data mining and machine learning techniques. Well-known classification methods that have been widely applied so far are no longer feasible in Big Data environments. For this reason, prototype reduction techniques (both selection and generation) come up as a candidate solution to build a reduced version of the dataset that speeds up the execution of algorithms such as k-Nearest Neighbors and overcome their memory constraints. However, these solutions generally have a quadratic $O(N^2)$ time complexity and share similar limitations to those encountered in data mining and machine learning algorithms in terms of time and memory requirements. In order to overcome these limitations, we introduce a new distributed MapReduce prototype generation method called CHI-PG that provides a linear $O(N)$ time complexity and ensures constant accuracy regardless of the degree of parallelism. This approach builds prototypes by applying a simple scheme based on the rule generation process of the Chi et al. Fuzzy Rule-Based Classification System and takes advantage of the suitability of this classifier for the MapReduce paradigm. The empirical study shows that our new approach significantly improves the execution time of a state-of-the-art distributed prototype reduction algorithm (MRPR) without decreasing (and even improving) classification accuracy and reduction rates. Moreover, CHI-PG has been shown to be a candidate solution to the time and memory constraints of k-Nearest Neighbors when tackling large-scale datasets.

Cover Letter

Departamento de Automatica y Computacion
Universidad Publica de Navarra
Pamplona, Spain 31006
Monday, 24th July 2017

Dear Prof. Zidong Wang and Steven Hoi

Please find enclosed a manuscript entitled “CHI-PG: A fast prototype generation algorithm for Big Data classification problems” which I am submitting for exclusive consideration of publication as an article in *Neurocomputing*. This manuscript is the authors' original work and has not been published nor has it been submitted simultaneously elsewhere. All authors have checked the manuscript and have agreed to the submission.

This paper presents a new distributed MapReduce Prototype Reduction approach with linear time complexity called CHI-PG. This method overcomes the usual quadratic time complexity barrier encountered in most PR methods, which makes it capable of dealing with Big Data problems. Furthermore, the performance of this approach (in terms of classification accuracy and reduction rate) is always the same regardless of the degree of parallelism used for the execution.

Thank you for your consideration of our work. Please address all correspondence concerning this manuscript to me at the Public University of Navarre and feel free to correspond with me by e-mail (mikel.elkano@unavarra.es).

Sincerely,

Mikel Elkano.

CHI-PG: A fast prototype generation algorithm for Big Data classification problems

Mikel Elkano[1,2], Mikel Galar[1,2], Jose Sanz[1,2], Humberto Bustince[1,2]

[1]Departamento de Automatica y Computacion, Universidad Publica de Navarra, 31006 Spain

[2]Institute of Smart Cities, Universidad Publica de Navarra, 31006 Spain

Abstract

The growing amount of available data has become a serious challenge to data mining and machine learning techniques. Well-known classification methods that have been widely applied so far are no longer feasible in Big Data environments. For this reason, prototype reduction techniques (both selection and generation) come up as a candidate solution to build a reduced version of the dataset that speeds up the execution of algorithms such as k-Nearest Neighbors and overcome their memory constraints. However, these solutions generally have a quadratic O(N^2) time complexity and share similar limitations to those encountered in data mining and machine learning algorithms in terms of time and memory requirements. In order to overcome these limitations, we introduce a new distributed MapReduce prototype generation method called CHI-PG that provides a linear O(N) time complexity and ensures constant accuracy regardless of the degree of parallelism. This approach builds prototypes by applying a simple scheme based on the rule generation process of the Chi et al. Fuzzy Rule-Based Classification System and takes advantage of the suitability of this classifier for the MapReduce paradigm. The empirical study shows that our new approach significantly improves the execution time of a state-of-the-art distributed prototype reduction algorithm (MRPR) without decreasing (and even improving) classification accuracy and reduction rates. Moreover, CHI-PG has been shown to be a candidate solution to the time and memory constraints of k-Nearest Neighbors when tackling large-scale datasets.

Neurocomputing

CHI-PG: A fast prototype generation algorithm for Big Data classification problems

M. Elkano, M. Galar, Jose Sanz, Humberto Bustince

Research Highlights

- We present a new distributed MapReduce Prototype Reduction method.
- This approach features a linear time complexity.
- The prototypes are built using fuzzy rules.
- The number of Mappers/Reducers does not affect the accuracy of the model.
- The main advantages are flexibility and simplicity.

Neurocomputing

Re: NEUCOM-D-17-02059

CHI-PG: A fast prototype generation algorithm for Big Data classification problems

Mikel Elkano, Mikel Galar, Jose Sanz, Humberto Bustince.

Authors' Response letter

First of all we would like to thank the reviewers for reading and revising our work. The paper has been carefully revised, following the referees' recommendations.

In what follows, we discuss the details on how the manuscript has been modified to address the concerns raised in the reviews.

Main Changes as suggested by reviewers:

- As suggested by Reviewer #1, the abstract has been shortened from 278 to 232 words.
- As suggested by Reviewer #4, the paper is now more concise (from 28 to 25 pages and from 11,695 to 10,375 words). Modified sections are:
  - 1. Introduction: less details about the methods are given.
  - 2. Preliminaries: old "Section 3. Related Work" has been shortened and moved to "Section 2.3 Related work".
  - Old Section 3. Related Work (new Section 2.3 Related work): MRPR's subsection does not exist anymore. The details about this method have been included in "Section 2.3 Related work" and "Section 4.2. Parameters setup".

REVIEWER #1

RV: Reviewer

AA: Author Answers

RV:

This paper proposed "CHI-PG", a fuzzy-rule-based prototype generation algorithm that is suitable for map-reduce and hence valuable in big data problems.

It's called "CHI-PG" because the fuzzy rule is inspired by the Fuzzy Rule-Based Classication System by Chi et al.. And PG means prototype generation.

The paper shows creative originality - combining fuzzy rules with PG with map-reduce.

It looks very sound technically, with exhaustive details of the algorithm and of the experiment. The proposed algorithm has a complexity to O(N), whereas competitors have ~O(N^2).

It's well written and easy to read.

It is presented in high quality with all the plots, tables, equations, and fonts neatly detailed.

My only suggestion (optional) is the abstract could be shorter.

AA:

We would like to thank the reviewer for reading and revising our work and for his/her comments. As suggested, the abstract has been shortened from 278 to 232 words.

REVIEWER #2

RV: Reviewer

AA: Author Answers

RV: This paper proposes a distributed MapReduce prototype generation method called CHI-PG that provides a linear O(N) time complexity. Related work is comprehensively summarized, the experiments are also carefully designed. The overall paper is well-written, I recommend it to be accepted as the journal paper.
AA: We want to thank the reviewer for reading and revising our paper and for his/her interest in it.

REVIEWER #4

RV: Reviewer

AA: Author Answers

RV: This paper builds on the author's previous work on leveraging the fuzzy rule-based classification system (FRBCS) in MapReduce paradigm for big data classification problems. The focus of this paper is to extend the same technique in solving prototype generation problem. The authors make original contributions by designing the MapReduce version FRBCS-based prototype generation algorithm and by conducting extensive experiments to compare this algorithm with a benchmark MapReduce prototype generation algorithm which was published in this journal in 2015. Strengths: 1. The authors creatively apply FRBCS in MapReduce prototype generation and articulate the problem and the solution well. 2. The work is built on existing work and thoroughly compared with a state of the art benchmark algorithm, showing improvements in performance. 3. Substantial and systematic experimental work are done to verify the effectiveness and efficiency of the proposed

algorithm, as well as its usefulness in a larger context.

Weaknesses:

1. One major limitation of applying FRBCS in big data classification is that they do not scale with the number of variables, because number of rules will explode as number for variables increases. The authors admittedly touched this point at the end of the paper.

2. The datasets used in this paper have maximum 41 variables, which are not very representative of nowadays big data classification problems. Perhaps this is due to the previous point.

3. The paper could be more concise. It's currently a little too long and verbose.

4. There are some typos of the paper, e.g. Page 12. "Section II.B." is a wrong reference. Page 20. 5.3 "asses" typo.

Overall, I think this paper has original contribution and is relevant to the field, therefore can be accepted after the authors made the revisions.

AA:

We would like to thank the reviewer for reading and revising our work and for his/her comments.

Regarding points 1-2, as commented by the referee, we mentioned the weaknesses of our method in the concluding remarks and consequently included a future research line regarding this issue.

As suggested in point 3, the paper is now more concise (from 28 to 25 pages and from 11,695 to 10,375 words). Modified sections are:

- 1. Introduction: less details about the methods are given.
- 2. Preliminaries: old “Section 3. Related Work” has been shortened and moved to “Section 2.3 Related work”.
- Old Section 3. Related Work (new Section 2.3 Related work): MRPR’s subsection does not exist anymore. The details about this method have been included in “Section 2.3 Related work” and “Section 4.2. Parameters setup”.

Finally, following point 4, we have checked the paper again and corrected some few typos and errors.

CHI-PG: A fast prototype generation algorithm for Big Data classification problems

Mikel Elkano[a,b], Mikel Galar[a,b], Jose Sanz[a,b], Humberto Bustince[a,b]

[a] *Departamento de Automatica y Computacion, Universidad Publica de Navarra, 31006 Spain*
[b] *Institute of Smart Cities, Universidad Publica de Navarra, 31006 Spain*

Abstract

The growing amount of available data has become a serious challenge to data mining and machine learning techniques. Well-known classification methods that have been widely applied so far are no longer feasible in Big Data environments. For this reason, prototype reduction techniques (both selection and generation) come up as a candidate solution to build a reduced version of the dataset that speeds up the execution of algorithms such as k-Nearest Neighbors and overcome their memory constraints. However, these solutions generally have a quadratic $O(N^2)$ time complexity and share similar limitations to those encountered in data mining and machine learning algorithms in terms of time and memory requirements. In order to overcome these limitations, we introduce a new distributed MapReduce prototype generation method called CHI-PG that provides a linear $O(N)$ time complexity and ensures constant accuracy regardless of the degree of parallelism. This approach builds prototypes by applying a simple scheme based on the rule generation process of the Chi et al. Fuzzy Rule-Based Classification System and takes advantage of the suitability of this classifier for the MapReduce paradigm. The empirical study shows that our new approach significantly improves the execution time of a state-of-the-art distributed prototype reduction algorithm (MRPR) without decreasing (and even improving) classification accuracy and reduction rates. Moreover, CHI-PG has been shown to be a candidate solution to the time and memory constraints of k-Nearest Neighbors when tackling large-scale datasets.

Key words: Prototype Reduction, Prototype Generation, Big Data, MapReduce, Fuzzy Rule-Based Classification Systems

1. Introduction

The concept of *Big Data* is becoming more and more popular as industry and science increasingly generate and manipulate vast amounts of raw data. Despite the wide range of definitions of the term Big Data [1, 12, 28, 34], it is generally referred to as those situations where the amount of data to be processed

Email addresses: mikel.elkano@unavarra.es (Mikel Elkano), mikel.galar@unavarra.es (Mikel Galar), joseantonio.sanz@unavarra.es (Jose Sanz), bustince@unavarra.es (Humberto Bustince)

exceeds the capacity of commodity computers in terms of time consumption and/or memory requirements. In this scenario, the need for adapting and redesigning well-known data mining and machine learning algorithms emerges [10, 30]. One of the most popular methodologies to deal with Big Data environments is *distributed computing*, where the data is distributed along multiple nodes of a *cluster*. In this manner, each node is only responsible for the processing of the split of data stored in it, and thus the processing of the whole dataset is parallelized along several nodes. There are many different solutions to implement this methodology and deploy the distributed system. *Apache Hadoop* [35] is a well-known open-source alternative, which provides a transparent distributed system with fault-tolerant mechanisms. This framework is based on an open-source implementation of the *MapReduce* algorithm [8] and a distributed file system called *Hadoop Distributed File System* (HDFS) [29].

Besides distributed computing, reduction techniques help existing algorithms to speed up the execution, decrease memory constraints, and even improve accuracy by cleaning and simplifying raw data [15]. In this work we focus on Prototype Reduction (PR) techniques [14, 27, 31, 32], which try to generate a reduced version of the original training set by building prototypes that replace the original instances when running instance-based classification algorithms, such as k-Nearest Neighbors (k-NN) [26]. PR solutions are divided into two main groups: Prototype Selection (PS) [14] and Prototype Generation (PG) [31]. In PS techniques, the reduced set contains the most representative instances selected from the original training set, whereas PG methods provide a reduced set by generating new instances based on the original ones.

Paradoxically, although data reduction techniques allow one to run traditional data mining and machine learning algorithms in large-scale problems, they share similar limitations to those encountered in these types of algorithms in terms of execution time and memory requirements. Most of the existing PR approaches have at least an $O(N^2)$ time complexity (where N is the number of instances), which makes them too time-consuming to face Big Data problems. Trying to overcome these constraints, Triguero et al. presented a new distributed solution called MRPR that allows existing PR techniques to be applied in large datasets [33]. However, the main drawback of this methodology is that the quadratic time complexity of most PR techniques is still inherited when the increase in the data size is much higher than that in the number of computing units, making this solution unfeasible for these cases. Moreover, the performance of this method may gradually drop as the degree of parallelism increases, accentuating this limitation.

In this paper, we present a new distributed PG approach with linear $O(N)$ time complexity called CHI-PG that overcomes the quadratic $O(N^2)$ time complexity barrier encountered in most PR methods. CHI-PG is a new PG technique specifically designed for Big Data classification problems. This approach emerged from the idea of considering the fuzzy rules obtained by the CHI-BD Fuzzy Rule-Based Classification System (FRBCS) [9] as prototypes, avoiding the most expensive part of the algorithm, i.e., the rule weight computation process. CHI-BD was presented in our previous work as a distributed solution to recover the original Chi et al. [6] FRBCS in Big Data problems. In this FRBCS, each training example generates

a new fuzzy rule without taking the rest of the examples into consideration (except for the rule weight computation), which makes the rule generation process easily parallelizable using MapReduce. The main features and novelties of our new PG model can be summarized as follows.

- The prototypes provided by CHI-PG are exactly the same regardless of the number of Mappers/Reducers used for its execution. Thus, the accuracy and reduction rate of the algorithm do not drop with the degree of parallelism.
- The entire reduction process requires only a single pass over the whole training set, ensuring a linear $O(N)$ time complexity.

In order to assess the quality of our proposal, we have developed an empirical study using 4 different large-scale datasets from the UCI repository [21]. In this study, we analyze both the performance and the time complexity of the proposed algorithm and perform a comparison between CHI-PG and MRPR. Additionally, the usefulness of CHI-PG for k-NN in large-scale problems is evaluated. The experimental results show that the prototype generation process carried out by CHI-PG is notably faster than the one performed by MRPR, maintaining and even improving the reduction rate and the classification accuracy of k-NN. Moreover, our proposal has been shown to be a candidate solution to allow k-NN to be applied in large-scale datasets where the usage of the entire training set is unfeasible. The full source code of CHI-PG is publicly available at GitHub (*https://github.com/melkano/chi-pg*) repository under the GNU General Public License.

This paper is organized as follows. Section 2 describes the basics of the Apache Hadoop framework and the Chi et al. FRBCS and summarizes some state-of-the-art PR approaches. In Section 3 we introduce our new distributed PR approach. Sections 4 and 5 present the experimental framework and the analysis of the empirical results, respectively. Finally, Section 6 concludes this paper.

2. Preliminaries

In this section, we outline the components and functioning of Apache Hadoop (Section 2.1) and describe some basic concepts about the Chi et al. FRBCSs (Section 2.2). Additionally, some recent PR methods are presented in Section 2.3.

2.1. Apache Hadoop

Apache Hadoop [35] (shortened to Hadoop) is an open-source Big Data framework that provides a transparent distributed system with fault-tolerant mechanisms. This transparency allows the user to devote all the time and effort to the design of the data processing stage. In order to provide this infrastructure, Hadoop is primarily based on two works published by Google: the *Google File System* (GFS) [16] and the

MapReduce algorithm [8]. In this manner, the core of Hadoop consists of a distributed file system based on the GFS called *Hadoop Distributed File System* (HDFS) [29] (storage part) and an implementation of MapReduce (processing part):

- *HDFS*: provides the storage layer of the system. The stored dataset is split into multiple physical blocks that are distributed among all the nodes of the cluster. When the dataset is processed, Hadoop follows a data locality policy transferring the code of the algorithm to all nodes instead of moving the data. In this manner, a node will preferably process the blocks stored in it (although a block stored in another node can be processed as well), avoiding heavy data transfers.

- *MapReduce*: brings the programming paradigm used for the distributed data processing, which is composed of the two following stages.

  1. *Map phase*: the first step consists in dividing the input data into multiple logical splits associated with different physical blocks (preferably with local ones, due to the data locality policy). Then, each split will be processed by a single processing unit called *Mapper* (although the user can assign multiple splits to a single Mapper). Each node can execute multiple Mappers concurrently. The input data received by the Mapper is first transformed into <k, v> key-value pairs that are processed by the *map*() function (defined by the user), which is called for each pair. This function returns another <k', v'> key-value pair that conforms the so-called *intermediate data*. Finally, this intermediate data is prepared to be sent to the Reducers by executing the following phases:
     (a) Sorting and Merging: key-value pairs are sorted by key and a list is generated for each key containing all its values (<k', list (v')>).
     (b) Partitioning: a target Reducer is selected for each key.
     (c) Shuffle: previous intermediate data is transferred to the Reducers.
  2. *Reduce phase*: the key-value pairs (<k', list (v')>) generated by the Mappers are aggregated by the Reducers. To this end, when all the Mappers have finished, the *reduce*() function (defined by the user) is called for every single key (k'), where all its values (list (v')) are aggregated. Finally, the Reducer produces the final result (v") for each key.

  Fig. 1 shows the data flow of the MapReduce algorithm. In addition to the *map* and *reduce* phases, there is an optimization for MapReduce jobs called *Combiner*. This component is locally executed on the output of the *map* phase (<k', list (v')>) and represents a local mini-Reducer that minimizes the intermediate data sent to the Reducers by previously aggregating the values of each key (list(v')). In fact, the Combiner and the Reducer share the same code in several cases.

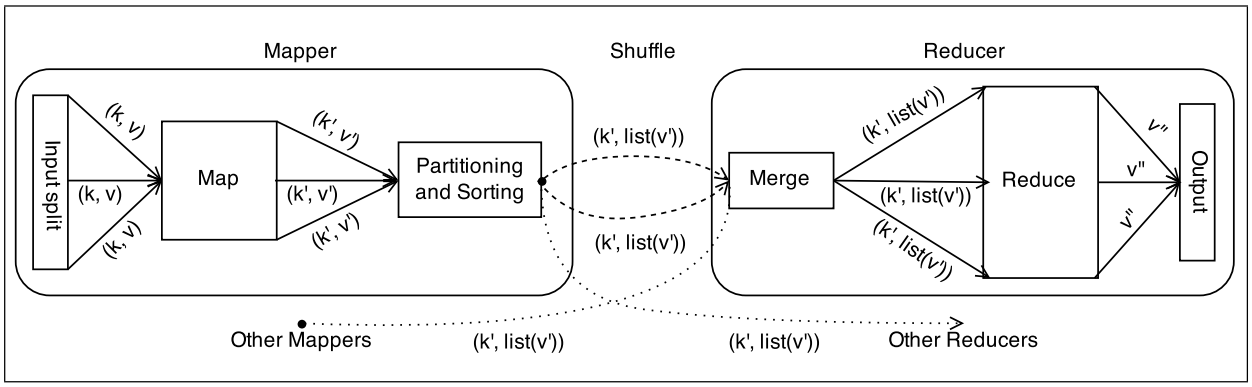


Figure 1: MapReduce data flow.

2.2. Fuzzy Rule-Based Classification Systems: Chi et al. algorithm

Since our approach is based on the idea behind the rule learning process carried out by the Chi et al. FRBCS [6], in this section we recall some basic concepts about FRBCSs and describe the learning algorithm of the Chi et al. method.

FRBCSs are well-known and widely used tools in the field of pattern recognition and machine learning. These systems provide an interpretable model by generating human-readable rules composed of linguistic labels [18], using the following structure:

$$\text{Rule } R_j: \text{ If } x_1 \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_D \text{ is } A_{jD} \text{ then Class} = C_j \text{ with } RW_j \tag{1}$$

where R_j is the label of the j-th rule, $x = (x_1, \ldots, x_D)$ is a D-dimensional pattern vector that represents an example, A_{ji} is a linguistic label, C_j is the consequent (label of the class), and RW_j is the rule weight. In this work, linguistic labels are represented by fuzzy sets modeled by triangular shaped membership functions (Fig. 2). Regarding the weight of a given rule, it represents the confidence degree of the rule and has a significant impact on the performance of FRBCSs [17]. A number of different approaches have been proposed in the literature in order to compute rule weights, such as the commonly used Certainty Factor and Penalized Certainty Factor [19], or the more recent Penalized Cost-Sensitive Certainty Factor [22].

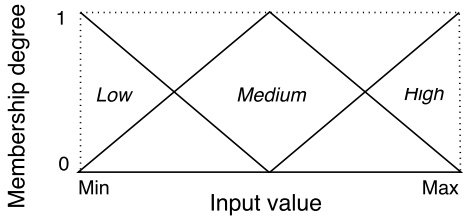


Figure 2: Example of a fuzzy variable with 3 fuzzy sets (linguistic labels) modeled by triangular membership functions.

In order to build the rule base, the original Chi et al. algorithm applies the following learning process.

1. *Construction of the linguistic labels.* The fuzzy sets (linguistic labels) are built with the same triangular shape and equally distributed on the range of values for each variable. All these fuzzy sets will be shared by all the rules.
2. *Generation of a fuzzy rule for each example.* A fuzzy rule is generated for each training example x_p (with $p \in \{1, \ldots, P\}$, where P is the number of training examples) as follows.
   (a) The membership degrees of each value x_{pd} ($d \in \{1, \ldots, D\}$) to all the different fuzzy sets of the d-th variable are computed.
   (b) For each variable, the linguistic label with the greatest membership degree is selected.
   (c) The antecedent part is determined by the intersection of the selected linguistic labels and the consequent is the class label of the example (y_p). All the rules will have exactly the same number of antecedents as number of variables in the problem (D).
   (d) The fuzzy rule is added to the rule base if it is not already included.
   (e) The rule weight is computed applying one of the above-mentioned approaches.
3. As we can observe, conflicting rules can be obtained, that is, rules with the same antecedent part and different consequent (class). In these cases, only the rule having the highest weight is kept in the rule base whereas the remainder ones are removed. Furthermore, rules having a negative weight are not generated.

The fact that each training example generates a new rule without taking the rest into account allows this algorithm to take advantage of the power of MapReduce. The entire learning process is directly parallelizable in the *map* stage until the computation of rule weights. Leveraging this feature, we proposed a global distributed solution for Big Data classification problems called CHI-BD [9] that recovers the original Chi et al. FRBCS with no approximations. CHI-BD performs a rule generation process composed of two stages:

- Generation of candidate fuzzy rules: a new rule is built for each example.
- Computation of rule weights: the weight of each rule is computed and duplicates/conflicts are resolved.

The prototype generation method proposed in this work (CHI-PG) is based on the first stage of the rule generation process of CHI-BD. Nonetheless, we must stress that CHI-PG makes use of fuzzy rules to create prototypes instead of a FRBCS, given that the algorithm presented in this work is not a classifier.

2.3. Related work

The purpose of Prototype Reduction (PR) techniques [27] is to provide a reduced version of the training set that maximizes the classification accuracy for unseen data while minimizing the number of instances in the reduced set [20]. There are two different approaches for PR, depending on whether prototypes are selected [14] or generated [31], or both models are combined [32]. Even though many proposals have

successfully dealt with PR problems [27], the scalability of these solutions when tackling large datasets is still a major constraint. The two main reasons are time complexity and memory requirements. With respect to the former, the complexity of most PR models is usually at least $O(N^2)$ [15]. Regarding the memory consumption, in general the entire training set needs to be loaded into the main memory along with several data structures used to store partial computations, which may exceed the available memory. All these drawbacks reveal the need for exploring new distributed approaches in order to tackle the increasingly large datasets.

The approach given by Cano et al. in [3] was one of the first published methods aimed at overcoming this limitation. In this work, the authors present a stratification strategy that splits the training set into several disjoint subsets with equal class distribution. Then, any of the existing PS algorithms is separately applied in each subset and all the selected instances are combined. Considering a PS algorithm of quadratic time complexity $O(N^2)$ (N being the number of instances) and taking T subsets, the final time complexity will be $O(N^2/T^2)$ when running in parallel and $O(N^2/T)$ in the sequential mode, obtaining a speedup of T^2 and T, respectively. Following this methodology, a number of methods were later proposed in a series of works [4, 13, 32, 33]. In [4], an evolutionary PS method is applied using stratification in order to improve the trade-off between accuracy and interpretability in C4.5. Garcia et al. [13] presented a scaling up approach of a memetic algorithm (SSMA) that improves the performance of the previous evolutionary PS methods and takes advantage of stratification to scale up. In [32], the authors introduced a hybrid approach called SSMA-SFLSDE that combines PS and PG techniques to obtain the reduced training set.

However, the aforementioned approaches inherit remarkable drawbacks from the stratification strategy presented by Cano et al. in [3]. Firstly, this method is likely to suffer performance loss (in terms of classification accuracy and reduction rate) as the degree of parallelism (number of subsets) increases. This is due to the fact that each local PR process performs a local optimization without considering the remaining subsets, and thus it strongly depends on the distribution and the number of subsets. Secondly, the stratification process requires the entire training set to be stored in the main memory, and hence this strategy does not scale to datasets of arbitrary size. Finally, the direct joining of all the reduced sets obtained from the different subsets may result in noisy and/or redundant instances.

In order to provide a solution to these issues, Triguero et al. presented a MapReduce approach (MRPR) [33] that applies a divide and conquer strategy to run existing PR techniques in large datasets. To this end, each Mapper runs a local reduction process in the corresponding data partition using a certain PR method, obtaining one reduced set per mapper. Next, all these partial sets are aggregated by the Reducer removing noisy and redundant instances. The authors tested several PR methods for the Map stage (including the aforementioned SSMA-SFLSDE [32], which was the best performing one) and three strategies for the Reduce stage (join, filtering, and fusion). Although this methodology resolves the issues associated with memory constraints and noisy/redundant instances, the performance of MRPR may drop as the number of Mappers

increases, and thus it is still dependent on the degree of parallelism. Since this method is included in the experimental study carried out in this work, more details about MRPR are given in Section 4.2.

In addition to this series of contributions, some promising solutions have been published to deal with the scalability problems of PR techniques. A distributed approach for scaling up any PS algorithm by means of a voting scheme was presented by de Haro-García et al. in [7]. In this method, the original training set is divided into multiple disjoint subsets (of a fixed size given by the user) and a PS algorithm is concurrently applied in each subset, where the instances that are selected to be removed receive a vote. Unlike in the aforementioned stratification strategy, this process is repeated several times (called rounds of stratification) in order to minimize the effect of the random partitioning. Finally, the different rounds are combined applying a voting scheme, where the instances that have received a higher number of votes than a given threshold are discarded. In [24], the authors proposed converting the original training set into an undirected and weighted k-NN graph (where each node represents a data point and each edge represents the similarity between the data points) to apply the *Fast and Unique Representative Subset selection* (FURS) technique [25]. The Column Subset Selection (CSS) problem [2] can be considered as a generalization of both the instance selection and the unsupervised feature selection problems. Farahat et al. introduced a distributed CSS algorithm [11] that can be applied to select representative instances (or features) in large datasets.

Nevertheless, almost none of the previously mentioned contributions have achieved a linear time complexity in large-scale datasets, reaching an $O((N/m)^2)$ at best in most cases, where m is the number of cores used for the execution. This is due to the fact that all of them inherit the usual $O(N^2)$ time complexity of PR techniques when the increase in the data size is much higher than that in the degree of parallelism. The lowest time complexity among the aforementioned methods is probably the one presented by de Haro-García et al. [7]. The total time complexity of this algorithm is $O((1/p) \cdot r \cdot (N/s) \cdot K)$, where K is the number of operations needed by the PS algorithm in a subset of size s, N is the number of instances in the original training set, r is the number of rounds, and p is the number of processors. The authors state that an overall linear time complexity (with respect to the number of instances) is achieved. This is due to the fact that, although the time complexity of the PS algorithm (K) is usually quadratic (at best) with respect to the subset size (s), K is constant (because s has been previously fixed by the user) and s is small enough to allow the PS method to run in a short period of time. However, when the increase in the number of instances (N) is much higher than that in the number of processors (p), if the size of each subset (s) is small, the number of subsets to be sequentially processed increases (because there are more subsets that cannot be concurrently processed by the PS algorithm), limiting the scalability of the model. In this same hypothetical scenario, if we take a too large s (aiming at parallelizing the processing of the subsets), the quadratic time complexity of the PS algorithm (reflected in K) may be a bottleneck.

3. CHI-PG: designing a distributed prototype generation algorithm with linear time complexity

In this paper, we present a new distributed PG method for Big Data classification problems called CHI-PG. This work was motivated by the need for designing linearly scalable PR methods to deal with large-scale datasets. Therefore, we try to address two important issues:

- To obtain a model with linear time complexity with respect to the number of instances.
- To maintain the accuracy of the algorithm regardless of the degree of parallelism.

3.1. CHI-PG algorithm

The idea behind CHI-PG is to take advantage of the suitability of the Chi et al. FRBCS for the MapReduce paradigm and the possibility of constructing prototypes from the rules generated by this algorithm. We have to point out that, in our method, we consider that a fuzzy rule is composed of the antecedents and the consequent (class) without the rule weight. That is, a fuzzy rule has the following structure:

$$R_j: \text{ If } x_1 \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_D \text{ is } A_{jD} \text{ then Class} = C_j.$$

The usage of triangular shaped membership functions in the antecedents of these rules (without weights) implies that the input space is split as a grid, which is composed of several *cells*. A cell is a region of the input space where the fuzzy sets, which delimit it, have the largest membership degree. In Fig. 3 we can observe an input space with two variables, which is split into 9 cells due to the fact that 3 fuzzy sets are used to model each variable. Each cell determines the area of the input space where the corresponding fuzzy rule is the most appropriate one among all the fuzzy rules that can be fired simultaneously. Thus, for the sake of simplicity, in this paper we will refer indistinctly to the terms antecedents of a fuzzy rule and the corresponding cell. The fact that these cells are disjoint allows us to develop a linear PG method.

However, as it can be seen in Fig. 3, examples belonging to several classes can fall in the same cell, that is, there can be rules with the same antecedents referring to the same cell but with different consequents (classes). In fact, these fuzzy rules are the conflicting rules as explained in Section 2.2. The basic idea of our method is based on the fact that each fuzzy rule can be seen as the representative of the examples falling in the corresponding cell belonging to the consequent class. We can denote the subset of examples represented by the j^{th} rule as $X_j = \{x_1, \ldots, x_{N_j}\}$, with $N_j = |X_j|$. One should notice that:

$$\bigsqcup_{i=1}^{J} X_j = X,$$

since each example will be represented by one rule (the one it generates), being J the number of rules in the rule base.

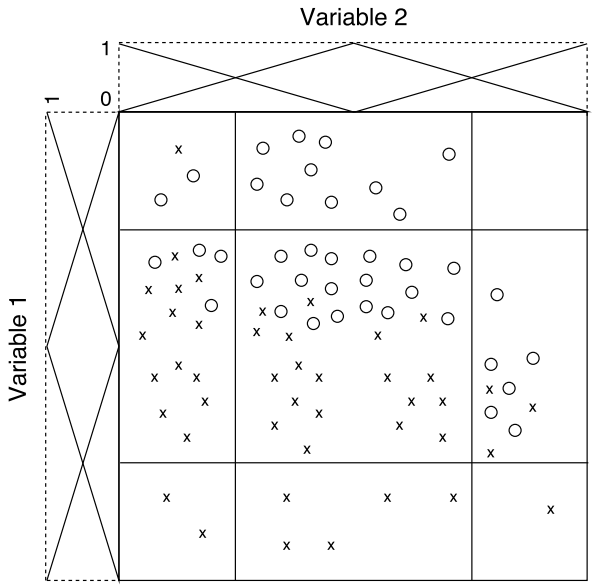


Figure 3: Cells in the feature space.

Therefore, we want to obtain the most representative prototypes for each cell using the rules that refer to it. In order to do so, our aim is to fuse examples represented by each rule generating one prototype per rule. That is, in each cell as many number of prototypes as rules referring to the cell are generated, and consequently J prototypes are finally obtained.

More specifically, the representative prototype for the j^{th} rule is obtained by computing the arithmetic mean of all the examples in X_j for each attribute. Notice that in the case of nominal attributes, the mode must be used, but for the sake of simplicity we focus our explanations on numerical attributes. Therefore, for rule j, the value for the d^{th} attribute of the new prototype $s_j = (s_{j1}, \ldots, s_{jD})$ is generated as follows.

$$s_{jd} = \frac{\sum_{i=1}^{N_j} x_{id}}{N_j}. \tag{2}$$

We name this method CHI-PG Arithmetic Mean (CHI-PG_AM). However, from our point of view, generating prototypes from different classes in the same cell could lead to obtaining unsuitable prototypes mainly when N_j is low. For this reason, we propose two possible techniques to deal with this problem.

1. In order to minimize the generation of prototypes from rules representing few examples, we can limit their generation by demanding a minimum number of represented examples ($MinEx$). That is, a prototype is generated only if $N_j > MinEx$.
2. However, even using the first solution the problem can remain when more than one rule exceed the threshold value $MinEx$. We can opt for trying to solve these conflicts as it is usually done in FRBCSs. In order to do so, we propose combining the thresholding with a second method named CHI-PG_SAM

(Single Arithmetic Mean), which solves the conflicts by only generating a prototype for each cell. The prototype to be generated is the one created from the rule with the largest N_j.

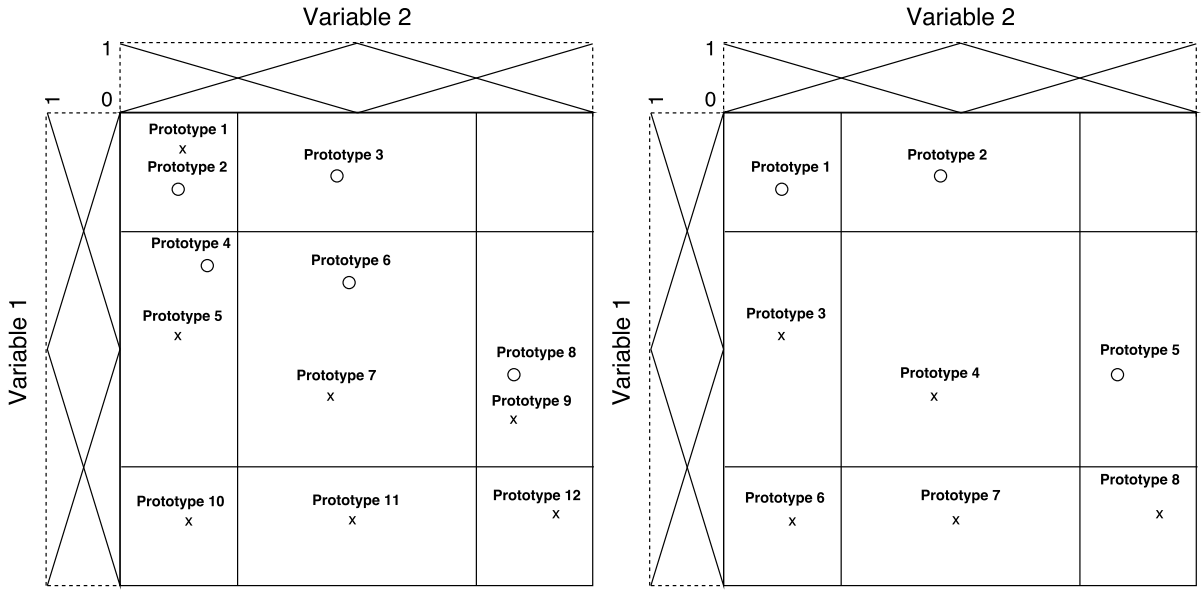
In the experimental study, we will analyze the behavior of both methods and the different values for their parameters. We have to stress that CHI-PG complexity is linear with respect to the number of examples, since each one can be independently considered to generate its rule, allowing the generation of all the prototypes to be carried out in a single pass over the whole training set (implementation details are given in the next Section).

In order to understand the proposed method, hereafter we provide an illustrative example. Consider the two-dimensional two-class problem shown in Fig. 3. The input space is divided into 9 cells, but given that more than one rule can be generated referring to the same cell, a total of 12 rules are generated. From these rules and its associated examples, new prototypes are generated. In the case of CHI-PG$_{AM}$ and considering $MinEx = 1$ (given the low number of examples), the resulting 12 prototypes can be observed in Fig. 4a. For instance, if we look at the upper-left most cell, we find a prototype for each class, since there are examples from both classes, whereas in the upper-middle cell we find a unique prototype because in that cell there are only examples from one class. Regarding CHI-PG$_{SAM}$, the resulting number of prototypes is 8, since a maximum of one prototype per cell is generated and there is one cell without examples. Fig. 4b depicts the prototypes obtained. The difference between both methods can be observed looking at the upper-left most cell, where, in this case, we only find one prototype (the one created from the rule with the largest N_j) instead of two.

3.2. MapReduce implementation

In order to distribute the prototype generation process, we make use of the first stage of the rule generation process carried out by CHI-BD [9]. Next, the details about the MapReduce implementation are given, whose pseudocode is shown in Algorithms 1-3.

1. *Map phase.* A new fuzzy rule is generated for each input example in the *map*() function (following the learning algorithm shown in Section 2.2) and sent to the combiner (Algorithm 1, Lines 1-2), where the sorting and merging phases group the instances represented by each rule by taking the antecedent part referring to the cell as the key. Then, the combiner counts the number of instances represented by each rule and adds the values of those instances (Algorithm 2, Lines 1-9), gathering all the information needed for the arithmetic mean computation shown in Eq. (2). We must remark that in the case of nominal variables, the sums of the values are replaced by the counters used for the mode computation. However, for the sake of simplicity, in this section we only show the implementation related to numerical variables. Finally, all this information is sent to the Reducer using $< antecedents, (consequent, (numInstances, partialSum)) >$ pairs (Algorithm 2, Lines 10-12), where



(a) CHI-PG$_{AM}$ (b) CHI-PG$_{SAM}$

Figure 4: Prototype generation in CHI-PG (minimum threshold for $N_j = 1$).

the key is the antecedent part referring to the cell and the value contains the consequent (class) and the count and sum of all the instances represented by each rule.

2. *Reduce phase.* All the partial counts/sums received from the Mappers are added, in such a way that the total count/sum of the instances represented by each rule is obtained. Since the sorting and merging phases automatically group the count and the sum of the instances represented by a certain rule, in each input key-value pair the Reducer has all the information required to generate a new prototype according to the selected method (AM or SAM).

   - *AM*: the Reducer creates a new prototype for each rule that represents a minimum number of instances. To do so, the arithmetic mean of all the instances represented by the rule is computed.
   - *SAM*: the prototype generation process is the same as in AM, but in this case, only the rule obtained by the highest number of instances is considered. Thus, for each key-value pair a maximum of one prototype will be created.

According to the aforementioned procedure, CHI-PG provides the following remarkable features:

- The reduced set obtained will be exactly the same regardless of the number of Mappers/Reducers used for its execution. This means that the accuracy and reduction rate of the algorithm do not decrease with the degree of parallelism.

- The entire prototype generation process only requires a single pass over the whole training set. Thus,

the linear time complexity is maintained with respect to the number of instances.

Algorithm 1 Mapper *map*() function in CHI-PG.

Function *map* (key, value)

Input: <*key*, *value*> pair, representing the class and the values of the instance, respectively.

Output: <*key'*, *value'*> pair, where *key'* represents the antecedents of the generated rule and *value'* is a pair <*classIndex*, (1, *instanceValues*)>.

Begin

1: *ruleAnts* ← *generateRuleFromInstance* (*value*)
2: EMIT (*ruleAnts*, <*getClassIndex*(*key*), (1, *value*)>)

End

Algorithm 2 Combiner *reduce*() function in CHI-PG.

Function *reduce* (key, values)

Input: <*key'*, *values*> pair, where *key'* represents the antecedent part referring to the cell and *values* is a list containing <*classIndex*, (*numInstances*, *sumValues*)> pairs that store the count and the partial sum of those instances falling in the cell *key'* for each represented class.

Output: <*key"*, *value"*> pair, where *key"* is equal to *key'* and *value"* is a list of pairs in the same format as in *values*.

Begin

{Add the values of all the instances belonging to each class and count the number of instances of each class}
{*classSumValues* is a vector containing the sum of the values for each class. For nominal variables, it is replaced by a counter used to compute the mode in the Reducer.}
{*classNumInstances* is a vector containing the number of instances for each class }

1: **while** *values.hasNext* () **do**
2: *currentPair* ← *values.next* ()
3: *currentClass* ← *currentPair.getClass* ()
4: *currentValues* ← *currentPair.getValues* ()
5: **for** *varIndex* = 1 **to** *NUM_VARIABLES* **do**
6: *classSumValues*[*currentClass*][*varIndex*] ← *classSumValues*[*currentClass*][*varIndex*] + *currentValues*[*varIndex*]
7: **end for**
8: *classNumInstances*[*currentClass*] ← *classNumInstances*[*currentClass*] + *currentPair.getNumInstances* ()
9: **end while**

{*listPairs* contains the count and the partial sum of all the instances belonging to each class.}

10: *listPairs* ← {}
11: **for** *classIndex* = 1 **to** *NUM_CLASSES* **do**
12: **if** *classNumInstances*[*classIndex*] ≥ 1 **then**
13: *listPairs.add*(<*classIndex*, (*classNumInstances*[*classIndex*], *classSumValues*[*classIndex*])>))
14: **end if**
15: **end for**

{Send the list of pairs.}

16: EMIT (*key'*, *listPairs*)

End

4. Experimental framework

In this section, the experimental framework used to carry out the experiments in Section 5 is presented. First, the features of the datasets considered for the experimental study are shown (Section 4.1). Next, the

Algorithm 3 Reducer *reduce*() function in CHI-PG.

Function *reduce* (key, values)

Input: <*key'*, *values*> pair, where *key'* represents the antecedent part referring to the cell and *values* is a list containing <*classIndex*, (*numInstances*, *sumValues*)> pairs that store the count and the partial sum of those instances falling in the cell *key'* for each represented class.

Output: <*key"*, *value"*> pair, where *key"* and *value"* are the values and the class of the prototype, respectively.

Begin

{Add the partial sum of all the instances belonging to each class and count the number of instances of each class. This step is the same as the one carried out in the Combiner.}

{*classSumValues* is a vector containing the sum of the values for each class. For nominal variables, it is replaced by a counter used to compute the mode.}

{*classNumInstances* is a vector containing the number of instances for each class }

1: **while** *values.hasNext* () **do**
2: *currentPair* ← *values.next* ()
3: *currentClass* ← *currentPair.getClass* ()
4: *currentValues* ← *currentPair.getValues* ()
5: **for** *varIndex* = 1 **to** *NUM_VARIABLES* **do**
6: *classSumValues*[*currentClass*][*varIndex*] ← *classSumValues*[*currentClass*][*varIndex*] + *currentValues*[*varIndex*]
7: **end for**
8: *classNumInstances*[*currentClass*] ← *classNumInstances*[*currentClass*] + *currentPair.getNumInstances* ()
9: **end while**

{Two variants of CHI-PG: CHI-PG$_{AM}$ and CHI-PG$_{SAM}$}

10: **if** *CHI-PG*$_{AM}$ **then**
11: **for** *classIndex* = 1 **to** *NUM_CLASSES* **do**

{If there are enough instances belonging to this class, compute the arithmetic mean of those instances and create a new prototype. For nominal attributes, compute the mode instead of the arithmetic mean.}

12: **if** *classNumInstances*[*classIndex*] ≥ *MinEx* **then**
13: **for** *varIndex* = 1 **to** *NUM_VARIABLES* **do**
14: *classSumValues*[*classIndex*][*varIndex*] ← *classSumValues*[*classIndex*][*varIndex*] / *classNumInstances*[*classIndex*]
15: **end for**
16: EMIT (*classSumValues*[*classIndex*], *getClassLabel*(*classIndex*))
17: **end if**
18: **end for**
19: **else**

{Get the majority class}

20: *majClass* ← *getMajorityClass* (*classNumInstances*)

{If there are enough instances belonging to the majority class, compute the arithmetic mean of those instances and create a new prototype. For nominal attributes, compute the mode instead of the arithmetic mean.}

21: **if** *classNumInstances*[*majClass*] ≥ *MinEx* **then**
22: **for** *varIndex* = 1 **to** *NUM_VARIABLES* **do**
23: *classSumValues*[*majClass*][*varIndex*] ← *classSumValues*[*majClass*][*varIndex*] / *classNumInstances*[*majClass*]
24: **end for**
25: EMIT (*classSumValues*[*majClass*], *getClassLabel*(*majClass*))
26: **end if**
27: **end if**

End

different methods used throughout the experiments are briefly described, along with the parameters setup selected for each of them (Section 4.2). Finally, we introduce the performance metrics that are necessary to assess the quality of our proposal (Section 4.3).

The full source code of CHI-PG is publicly available at GitHub (*https://github.com/melkano/chi-pg*) repository under the GNU General Public License.

4.1. Datasets

In order to carry out the experimental study, we have selected 4 different datasets containing more than a million instances from the UCI repository [21]. Table 1 depicts the features of the datasets showing the number of examples (#Examples), the number of real, integer, and nominal attributes (#Attributes), and the number of classes (#Classes). All the experiments have been developed applying a *5-fold stratified cross-validation scheme*. In this model, we randomly split the dataset into five partitions of data, each one containing 20% of the examples, and we employ a combination of four of them (80%) to train the system and the remaining one to test it. Therefore, the result for each dataset is computed as the average of the five partitions.

Table 1: Description of the datasets.

Dataset	Id	#Examples	#Attributes				#Classes
			Real	Integer	Nominal	Total	
Higgs	Higgs	11,000,000	28	0	0	28	2
KDDCup1999	KDD	4,898,431	26	0	15	41	5
Poker	Poker	1,025,009	0	10	0	10	10
Susy	Susy	5,000,000	18	0	0	18	2

4.2. Parameters setup

The two PR methods that have been tested in this study (MRPR and CHI-PG) are aimed at speeding up the classification process and reducing the storage requirements of k-NN [26]. This classifier finds the k closest instances in the training set (according to a certain distance metric) to the test pattern (neighborhood). The final prediction is made in favor of the most predominant class in the neighborhood. This algorithm belongs to the so-called *lazy learning* or *instance-based learning* category, since input examples are classified without a previous learning stage. Both the number of neighbors (k) and the distance metric are key parameters of this method. In this work, we have taken $k = 1$ (the most frequently used value for the evaluation of PR algorithms [14]), the euclidean distance, and we have applied our own distributed Spark [36] implementation of k-NN that provides exactly the same results regardless of the number of partitions (degree of parallelism) used for its execution. This implementation is similar to that proposed in [23].

Regarding MRPR [33], we remind this method applies a divide and conquer strategy to apply existing PR techniques in large-scale datasets. To this end, each Mapper runs a local reduction process in the corresponding data partition using a certain PR method to obtain a reduced set S_j. In this work, we consider the best performing PR algorithm according to [33], i.e., SSMA-SFLSDE [32]. Finally, all these partial reduced sets are aggregated by the Reducer removing noisy and redundant instances, obtaining the final reduced set S. The authors propose three different reduction strategies:

- *Join*: all the S_j sets are progressively concatenated into the final S. This is the simplest and fastest option, and thus it is included as the baseline in [33].
- *Filtering*: a filtering stage is carried out in order to get rid of noisy instances in S. For this purpose, edition-based prototype selection methods [14] are applied as the S_j sets are joined.
- *Fusion*: aiming at removing redundant instances from S, centroid-based prototype generation techniques [31] are successively applied during the formation of S. This reduction task is performed by merging similar instances [5].

Even though the fusion-based Reducer is the best performing strategy, in this work only the baseline aggregation method (join) is considered. The reason for selecting a single Reducer strategy is that the remaining ones (filtering and fusion) involve an extra prototype reduction stage which could also be applied in CHI-PG as well.

In the case of CHI-PG, we have tested the performance of several configurations corresponding to the following parameters:

- *Number of fuzzy sets* (FS): number of fuzzy sets (linguistic labels) considered for each variable. This parameter determines the total number of cells into which the feature space is divided, which equals FS^D, where D is the number of variables in the problem In this work, we have selected 2, 3, 4, 5, 7, and 9 fuzzy sets per variable.
- *Minimum number of examples per prototype* ($MinEx$): minimum number of examples represented by a rule (in a certain cell) required to generate a new prototype. In this study, we consider 3, 5, and 10 in the case of 2 and 3 fuzzy sets, 3 and 5 when taking 4 and 5 fuzzy sets, and 3 when considering 7 and 9 fuzzy sets. The reason for decreasing the minimum number of examples per prototype as the number of fuzzy sets per variable increases is that creating a large number of cells reduces the likelihood (a priori) of reaching such a minimum.
- *Reduce aggregation*: the aggregation considered in the *reduce* stage (AM or SAM).

With respect to the degree of parallelism considered for each algorithm, the number of Mappers/Reducers used in the experiments varies from method to method. The reason for selecting different degrees of par-

allelism depending on the algorithm is the huge differences existing among the time complexities, which makes the usage of a shared configuration unfeasible. In the case of CHI-PG, all the configurations use 32 Mappers and 8 Reducers for all the datasets, even though an optimal number of Reducers can speed up the execution of this method. On the other hand, we run MRPR using 64 (Poker) and 1024 (Higgs, KDDCup1999, Susy) Mappers. The criterion used to select these configurations for MRPR is the minimum number of Mappers that allows us to run the experiments in a reasonable time considering the data size (applying a 5-fold cross-validation) and the computing power of our cluster.

We must remark that all the methods have been configured on top of the same distributed environment and make use of the above-mentioned implementation of k-NN.

The infrastructure used to carry out the experiments is the following one. All the parallel methods have been executed in an 8 nodes cluster connected via 1Gb/s Ethernet LAN network. Half of these nodes are composed of 2 Intel Xeon E5-2620 v3 processors at 2.4 GHz (3.2 GHz with Turbo Boost) with 12 virtual cores in each one (where 6 of them are physical). Three of the remaining nodes are equipped with 2 Intel Xeon E5-2620 v2 processors at 2.1 GHz with the same number of cores as the previous ones. The last node is the master node, composed of an Intel Xeon E5-2609 processor with 4 physical cores at 2.4 GHz. All slave nodes are equipped with 32 GB of RAM memory, while the master works with 8 GB of RAM memory. With respect to the storage specifications, all nodes use Hard Disk Drives featuring a read/write performance of 128 MB/s. The entire cluster runs under CentOS 6.5 and Apache Hadoop 2.6.0. This configuration features up to 42 concurrent YARN containers, where each container can be a Mapper, a Reducer, or the Application Master.

4.3. Performance metrics

In order to assess the performance of the different methods, we first test the classification accuracy of k-NN when considering both the whole datasets and their respective reduced versions obtained by MRPR and CHI-PG. To this end, we apply the most common metric, that is, the *accuracy rate*:

$$Accuracy\ rate = \frac{Number\ of\ correctly\ classified\ examples}{Total\ number\ of\ examples}. \tag{3}$$

Additionally, we measure the *reduction rate* of each PR method defined as:

$$Reduction\ rate = \left(1 - \frac{Number\ of\ examples\ in\ the\ reduced\ set}{Number\ of\ examples\ in\ the\ original\ set}\right) \cdot 100. \tag{4}$$

All the execution times shown in the experimental study correspond to the total time needed by a given method to produce the results, including all the MapReduce/Spark stages (reading, writing, network communications, etc.). We must stress that the maximum number of YARN containers that can be concurrently executed in our cluster is of 42 (which equals to 41 Mappers/Reducers, considering the Application Master).

5. Experimental study

In this section, we test the performance of our approach (CHI-PG) by carrying out an empirical study composed of the following steps.

1. We evaluate the performance of the different configurations of CHI-PG proposed in this work. To this end, we analyze the classification accuracy obtained by k-NN when considering the reduced sets provided by each configuration, as well as the time needed to build the prototypes and the reduction rates (Section 5.1).
2. We assess the linear time complexity of CHI-PG by measuring the time consumption of different configurations and data sizes (Section 5.2).
3. We perform a comparison between CHI-PG and MRPR applying the aforementioned metrics, that is, classification accuracy, execution time, and reduction rate (Section 5.3).
4. We evaluate the usefulness of our approach for the k-NN classifier in large-scale datasets (Section 5.4). To this end, we compare the trade-off between the classification accuracy and the time needed to classify new patterns when considering both the whole training set and the prototypes provided by CHI-PG.

Tables 2, 3, and 4 show the results we analyze and evaluate in the first two steps of the study, that is, the average accuracy rate, the reduction rate, and the time needed by each PR method considered in this work, respectively. For each dataset, the best result is stressed in **bold-face**.

5.1. Analyzing the behavior of CHI-PG

The evaluation of a PR algorithm is usually performed by measuring the classification accuracy achieved by k-NN when considering the reduced set generated by the algorithm and the reduction rate reached with respect to the original training set. In these experiments, a number of different configurations have been applied to run CHI-PG, depending on the number fuzzy sets (FS) used for each variable, the minimum number of examples required to generate a new prototype ($minEx$), and the aggregation strategy considered (AM or SAM).

According to the average accuracy rates shown in Table 2, the SAM aggregation strategy provides better results than the AM, presumably because conflicts are resolved instead of generating prototypes that could be noisy. Regarding the influence of the $MinEx$ parameter (for the same number of fuzzy sets), we observe that it generally has a small impact on the classification accuracy, except in Poker, where in some cases an increase in $MinEx$ leads to a loss of classification accuracy, since some of the prototypes that may be necessary to correctly represent the instances are note generated. With respect to the number of fuzzy sets, the results obtained are strongly dependent on the dataset considered, although 4 fuzzy sets provide a robust overall performance.

Table 2: Accuracy rate obtained by k-NN when using the prototypes built by MRPR and CHI-PG.

Dataset	MRPR	CHI-PG$_{AM}$											
		2 FS			3 FS			4 FS		5 FS		7 FS	9 FS
		min3	min5	min10	min3	min5	min10	min3	min5	min3	min5	min3	min3
Higgs	57.20	55.58	55.60	55.66	56.83	57.03	57.45	57.88	**58.19**	55.34	55.76	47.39	47.01
KDD	99.54	99.42	99.44	99.47	99.78	99.80	99.80	99.86	99.87	99.87	99.89	99.93	**99.95**
Poker	51.17	44.18	47.97	51.39	52.54	**53.30**	47.43	52.48	49.84	50.65	46.25	48.79	46.05
Susy	69.21	65.13	65.29	65.71	67.30	67.49	67.74	68.34	68.70	68.43	69.25	69.23	**69.63**
Dataset	MRPR	CHI-PG$_{SAM}$											
		2 FS			3 FS			4 FS		5 FS		7 FS	9 FS
		min3	min5	min10	min3	min5	min10	min3	min5	min3	min5	min3	min3
Higgs	57.20	59.86	59.86	**59.87**	58.96	59.11	59.27	58.73	58.80	55.39	55.77	47.39	47.01
KDD	99.54	99.89	99.89	99.89	99.92	99.92	99.92	99.93	99.93	99.93	99.93	**99.94**	**99.94**
Poker	51.17	53.11	53.13	53.35	53.34	**53.50**	47.43	52.49	49.84	50.65	46.25	48.79	46.05
Susy	69.21	69.27	69.27	69.26	68.77	68.65	68.49	72.15	71.91	**73.02**	72.88	72.81	70.32

Table 3: Reduction rate achieved by MRPR and CHI-PG.

Dataset	MRPR	CHI-PG$_{AM}$											
		2 FS			3 FS			4 FS		5 FS		7 FS	9 FS
		min3	min5	min10	min3	min5	min10	min3	min5	min3	min5	min3	min3
Higgs	96.68	99.51	99.52	99.54	95.46	97.00	98.33	92.63	97.06	98.96	99.89	**99.98**	**99.98**
KDD	97.95	99.97	**99.98**	**99.98**	99.93	99.95	99.97	99.89	99.92	99.85	99.89	99.76	99.65
Poker	97.56	91.44	92.50	94.88	86.08	95.93	99.88	96.31	99.89	99.48	**99.99**	99.97	**99.99**
Susy	97.00	**99.99**	**99.99**	**99.99**	99.82	99.86	99.88	98.86	99.14	96.80	97.73	91.89	94.42
Dataset	MRPR	CHI-PG$_{SAM}$											
		2 FS			3 FS			4 FS		5 FS		7 FS	9 FS
		min3	min5	min10	min3	min5	min10	min3	min5	min3	min5	min3	min3
Higgs	96.68	99.75	99.76	99.76	96.95	98.07	98.97	94.08	97.69	99.00	99.89	**99.98**	**99.98**
KDD	97.95	99.97	**99.98**	**99.98**	99.94	99.95	99.97	99.89	99.92	99.86	99.90	99.76	99.65
Poker	97.56	95.81	95.84	96.38	88.11	96.19	99.88	96.32	99.89	99.48	**99.99**	99.97	**99.99**
Susy	97.00	**99.99**	**99.99**	**99.99**	99.88	99.91	99.93	99.15	99.39	97.67	98.42	93.54	94.94

Table 4: Time needed (mm:ss) for the prototype generation process in MRPR and CHI-PG.

Dataset	MRPR	CHI-PG$_{AM}$											
		2 FS			3 FS			4 FS		5 FS		7 FS	9 FS
		min3	min5	min10	min3	min5	min10	min3	min5	min3	min5	min3	min3
Higgs	3499:34	00:57	**00:53**	**00:53**	01:50	01:47	01:38	02:56	02:37	02:36	02:30	02:46	02:33
KDD	510:49	00:25	00:24	**00:23**	00:25	00:25	00:24	00:26	00:26	00:28	00:27	00:30	00:32
Poker	338:42	00:31	00:30	00:30	00:30	**00:29**	**00:29**	00:35	00:34	00:35	00:36	00:35	00:35
Susy	522:24	**00:23**	00:24	00:24	00:25	00:25	00:26	00:36	00:37	00:43	00:46	01:07	01:11
Dataset	MRPR	CHI-PG$_{SAM}$											
		2 FS			3 FS			4 FS		5 FS		7 FS	9 FS
		min3	min5	min10	min3	min5	min10	min3	min5	min3	min5	min3	min3
Higgs	3499:34	01:02	00:59	**00:54**	01:46	01:40	01:39	02:53	02:41	02:37	02:32	02:38	02:39
KDD	510:49	00:24	**00:23**	00:24	00:25	00:24	00:26	00:26	00:26	00:27	00:27	00:29	00:32
Poker	338:42	**00:30**	**00:30**	**00:30**	00:34	00:32	00:33	00:34	00:35	00:35	00:32	00:34	00:36
Susy	522:24	**00:24**	**00:24**	**00:24**	00:26	00:25	00:26	00:36	00:36	00:49	00:50	01:04	01:12

Looking at the reduction rates shown in Table 3, we observe that in all cases the SAM aggregation strategy produces less prototypes than the AM, as expected. Additionally, the less fuzzy sets we use, the higher the reduction rate will probably be, since the number of cells into which the input space is divided is lower. In the same manner, if the minimum number of examples required to generate a new prototype is high, the likelihood of obtaining high reduction rates increases.

Regarding the time needed by each configuration to produce the prototypes, Table 4 shows that a high number of fuzzy sets per variable generally implies an increase in the execution time, due to the fact that more rules are created and transferred through the network.

5.2. Assessing the linear time complexity

In order to analyze the time complexity of the proposed algorithm, we measure the time consumption of three different configurations: 2 FS, 4 FS, and 9 FS, all using $minEx = 3$, since low values of $minEx$ usually implies an increase in time consumption. We have selected these numbers of fuzzy sets in order to provide the results of the lowest, the medium, and the highest values used in the experiments for this parameter. Both Table 5 and Fig. 5 illustrate the time elapsed in seconds when using 10%, 20%, 40%, and 80% of Higgs.

According to the results, the time needed by the algorithm to build the prototypes increases linearly with respect to the data size. We should note that the case of 2 FS is not representative of the algorithm's behavior, since the number of prototypes generated becomes too low (this result would mean that our

Table 5: Total execution time (mm:ss) of CHI-PG in 10%, 20%, 40%, and 80% of Higgs.

Data size	2 FS	4 FS	9 FS
10%	00:31	00:43	00:42
20%	00:37	00:49	00:47
40%	00:45	01:21	01:13
80%	00:47	02:23	02:04

algorithm is even faster than the linear case). The real tendency is that shown by 4 FS and 9 FS, which is linear with respect to the dataset size.

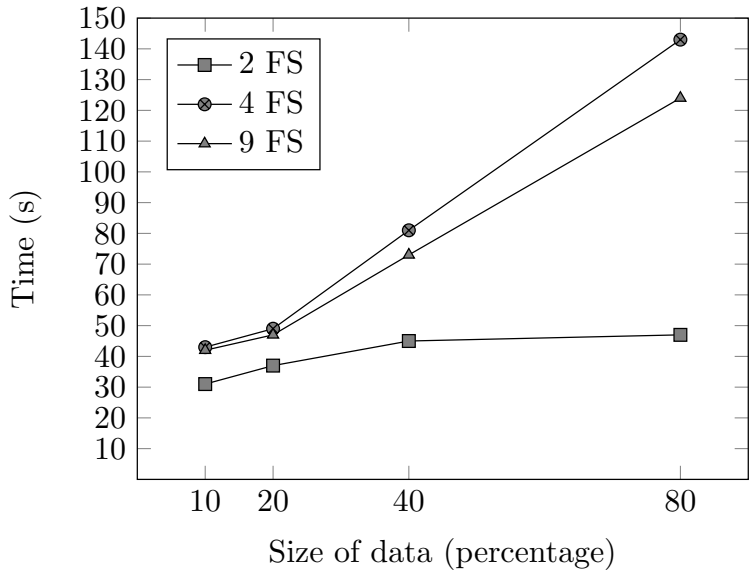


Figure 5: Time consumption (s) of different CHI-PG configurations.

5.3. Comparison between CHI-PG and MRPR

In order to compare CHI-PG and MRPR, we have used the same performance metrics as in the previous section, that is, classification accuracy, reduction rate, and execution time. As we can observe in Table 2, all the configurations of CHI-PG are able to maintain and even improve the average classification accuracy of MRPR. Regarding the reduction rate achieved in each method, Table 3 reveals that most of the configurations of CHI-PG provide a higher average reduction rate than that reached by MRPR.

The execution times extracted from the experiments (Table 4) clearly reveal the huge differences existing between $O(N^2)$ and $O(N)$ time complexities when tackling large-scale problems. In particular, in the case

of the largest datasets in terms of number of instances (Higgs, KDDCup1999, Susy), there is a difference of hours (or even days in Higgs) between both methods, in spite of using a higher degree of parallelism for MRPR. This means that in these cases the prototype generation process carried out by MRPR may be unfeasible due to the time constraints.

In summary, the experimental results have shown that CHI-PG is able to maintain (and improve in most cases) both the classification accuracy and the reduction rate obtained by MRPR while notably minimizing the time needed to build the prototypes. Moreover, the performance of CHI-PG is always the same regardless of the degree of parallelism, whereas in MRPR the performance will probably drop gradually as the number of Mappers increases.

5.4. On the usefulness of CHI-PG for large-scale k-NN

The usage of k-NN in large-scale datasets is seriously limited by its quadratic $O(N^2)$ time complexity and memory requirements. It is therefore worthwhile evaluating whether our approach could be a candidate solution to these limitations. To this end, we run our own distributed Spark implementation of k-NN considering both the whole training set and the prototypes built by CHI-PG (using the configuration providing the highest overall accuracy rate, that is, 4 fuzzy sets, a minimum of 3 examples per prototype, and SAM). Table 6 shows both the average accuracy rate and the time needed to classify the whole testing set in both cases. As we can observe, the classification accuracy obtained when using the prototypes provided by CHI-PG is promising in most cases, considering the remarkable improvement with respect to the time needed to classify the testing set. Moreover, looking at the reduction rates shown in Table 3, we notice that the usage of CHI-PG as a preprocessing tool might be a solution to the inability of commodity computers to load the whole training set in the main memory when facing huge datasets.

Therefore, our approach has been shown to be able of overcoming the time and memory constraints encountered in k-NN, revealing a promising trade-off between classification accuracy and time/memory consumption.

Table 6: Performance of k-NN when using both the whole training set and the prototypes built by CHI-PG.

Dataset	Average accuracy		Runtime (mm:ss)		Speedup
	Full	CHI-PG	Full	CHI-PG	
Higgs	**60.45**	58.73	5291:13	**160:11**	**33x**
KDD	**99.99**	99.93	94:30	**00:52**	**94x**
Poker	50.87	**52.49**	08:05	**00:42**	**8x**
Susy	69.34	**72.15**	28:22	**01:13**	**28x**

6. Conclusions

In this paper, we have introduced a new distributed MapReduce Prototype Generation (PG) algorithm called CHI-PG that provides a linear time complexity, while maintaining exactly the same performance regardless of the degree of parallelism. This method builds prototypes from the fuzzy rules obtained by the Chi et al. FRBCS and takes advantage of the suitability of this algorithm for the MapReduce paradigm. The results obtained in the experimental study reveal that the prototype generation process carried out by CHI-PG is remarkably faster than the one performed by MRPR, maintaining and even improving the reduction rate and the classification accuracy of k-NN. In addition to the comparison against MRPR, we have analyzed whether our approach is a candidate solution to the scalability problems of k-NN when facing Big Data classification problems. The experimental results show that our model allows one to run k-NN in datasets where the usage of the whole training set would make the execution of this classifier unfeasible, obtaining a promising classification accuracy.

In conclusion, the primary advantages of CHI-PG are flexibility and simplicity, allowing k-NN to be run in a wide range of Big Data classification problems in a much faster and more accurate way than one of the best performing state-of-the-art PR methods (MRPR). The main limitation of our approach is, however, that a steep increase in the number of variables would likely cause an explosive growth in the number of fuzzy rules generated by the Chi et al. FRBCS. This may lead to lower reduction rates and network bottlenecks, even though one could adjust the $MinEx$ parameter. In the future, the ability of our approach to deal with high-dimensional problems should be studied. A possible strategy to overcome the aforementioned drawbacks is to apply feature reduction techniques before using CHI-PG.

Acknowledgment

This work has been partially supported by the Spanish Ministry of Science and Technology under the project TIN2016-77356-P (AEI/FEDER, UE).

References

[1] Arthur, L., 2013. What is big data? Forbes.
URL http://www.forbes.com/sites/lisaarthur/2013/08/15/what-is-big-data/

[2] Boutsidis, C., Mahoney, M., Drineas, P., 2009. An improved approximation algorithm for the column subset selection problem. In: Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms. SODA '09. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 968–977.

[3] Cano, J., Herrera, F., Lozano, M., 2005. Stratification for scaling up evolutionary prototype selection. Pattern Recognition Letters 26 (7), 953 – 963.

[4] Cano, J., Herrera, F., Lozano, M., 2007. Evolutionary stratified training set selection for extracting classification rules with trade off precision-interpretability. Data & Knowledge Engineering 60 (1), 90 – 108, intelligent Data Mining.

[5] Chang, C.-L., 1974. Finding prototypes for nearest neighbor classifiers. IEEE Transactions on Computers C-23 (11), 1179–1184.

[6] Chi, Z., Yan, H., Pham, T., 1996. Fuzzy algorithms with applications to image processing and pattern recognition. World Scientific.

[7] de Haro-García, A., García-Pedrajas, N., del Castillo, J. R., 2012. Large scale instance selection by means of federal instance selection. Data & Knowledge Engineering 75, 58 – 77.

[8] Dean, J., Ghemawat, S., 2008. Mapreduce: Simplified data processing on large clusters. Commun. ACM 51 (1), 107–113.

[9] Elkano, M., Galar, M., Sanz, J., Bustince, H., 2017. CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems. Fuzzy Sets and SystemsIn Press.

[10] Fan, W., Bifet, A., 2013. Mining big data: Current status, and forecast to the future. SIGKDD Explor. Newsl. 14 (2), 1–5.

[11] Farahat, A., Elgohary, A., Ghodsi, A., Kamel, M., 2015. Greedy column subset selection for large-scale data sets. Knowledge and Information Systems 45 (1).

[12] Gandomi, A., Haider, M., 2015. Beyond the hype: Big data concepts, methods, and analytics. International Journal of Information Management 35 (2), 137 – 144.

[13] García, S., Cano, J., Herrera, F., 2008. A memetic algorithm for evolutionary prototype selection: A scaling up approach. Pattern Recognition 41 (8), 2693 – 2709.

[14] García, S., Derrac, J., Cano, J., Herrera, F., 2012. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (3), 417–435.

[15] García, S., Luengo, J., Herrera, F., 2014. Data Preprocessing in Data Mining. Springer Publishing Company, Incorporated.

[16] Ghemawat, S., Gobioff, H., Leung, S., 2003. The google file system. In: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles. SOSP '03. ACM, pp. 29–43.

[17] Ishibuchi, H., Nakashima, T., 2001. Effect of rule weights in fuzzy rule-based classification systems. IEEE Transactions on Fuzzy Systems 9 (4), 506–515.

[18] Ishibuchi, H., Nakashima, T., Nii, M., 2004. Classification and modeling with linguistic information granules: Advanced approaches to linguistic Data Mining. Springer-Verlag.

[19] Ishibuchi, H., Yamamoto, T., 2005. Rule weight specification in fuzzy rule-based classification systems. IEEE Transactions on Fuzzy Systems 13 (4), 428–435.

[20] Kuncheva, L. I., Galar, M., 2015. Theoretical and empirical criteria for the edited nearest neighbour classifier. In: 2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA, November 14-17, 2015. pp. 817–822.

[21] Lichman, M., 2013. UCI machine learning repository.
URL `http://archive.ics.uci.edu/ml`

[22] López, V., del Río, S., Benítez, M., Herrera, F., 2015. Cost-sensitive linguistic fuzzy rule based classification systems under the mapreduce framework for imbalanced big data. Fuzzy Sets and Systems 258 (0), 5 – 38.

[23] Maillo, J., Ramírez, S., Triguero, I., Herrera, F., 2017. kNN-IS: An iterative spark-based design of the k-nearest neighbors classifier for big data. Knowledge-Based Systems 117 (C), 3–15.

[24] Mall, R., Jumutc, V., Langone, R., Suykens, J., 2014. Representative subsets for big data learning using k-nn graphs. pp. 37–42.

[25] Mall, R., Langone, R., Suykens, J., 2013. FURS: Fast and unique representative subset selection retaining large-scale community structure. Social Network Analysis and Mining 3 (4), 1075–1095.

[26] Mclachlan, G. J., 2004. Discriminant Analysis and Statistical Pattern Recognition (Wiley Series in Probability and Statistics). Wiley-Interscience.

[27] Nanni, L., Lumini, A., 2011. Prototype reduction techniques: A comparison among different approaches. Expert Systems

with Applications 38 (9), 11820 – 11828.

[28] Philip Chen, C., Zhang, C.-Y., 2014. Data-intensive applications, challenges, techniques and technologies: A survey on big data. Information Sciences 275, 314–347.

[29] Shvachko, K., Kuang, H., Radia, S., Chansler, R., 2010. The hadoop distributed file system. In: Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). MSST '10. IEEE Computer Society, pp. 1–10.

[30] Suthaharan, S., 2014. Big data classification: Problems and challenges in network intrusion prediction with machine learning. SIGMETRICS Perform. Eval. Rev. 41 (4), 70–73.

[31] Triguero, I., Derrac, J., García, S., Herrera, F., 2012. A taxonomy and experimental study on prototype generation for nearest neighbor classification. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 42 (1), 86–100.

[32] Triguero, I., García, S., Herrera, F., 2011. Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification. Pattern Recognition 44 (4), 901 – 916.

[33] Triguero, I., Peralta, D., Bacardit, J., García, S., Herrera, F., 2015. MRPR: A MapReduce solution for prototype reduction in big data classification. Neurocomputing 150, Part A, 331 – 345.

[34] Wang, H., Xu, Z., Fujita, H., Liu, S., 2016. Towards felicitous decision making: An overview on challenges and trends of big data. Information Sciences 367-368, 747–765.

[35] White, T., 2009. Hadoop: The Definitive Guide, 1st Edition. O'Reilly Media, Inc.

[36] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., Stoica, I., 2010. Spark: Cluster computing with working sets. In: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing. HotCloud'10. USENIX Association, Berkeley, CA, USA, pp. 10–10.

5. CFM-BD: algoritmo distribuido de inducción de reglas para la construcción de Modelos Difusos Compactos en problemas de clasificación Big Data – *CFM-BD: a distributed rule induction algorithm for building Compact Fuzzy Models in Big Data classification problems*

El trabajo asociado a esta parte es:

- M. Elkano, M. Galar, J. Sanz, E. Barrenechea, H. Bustince "CFM-BD: a distributed rule induction algorithm for building Compact Fuzzy Models in Big Data classification problems", Enviado a *IEEE Transactions on Fuzzy Systems*, 2017.
    - Estado: Enviado.
    - Índice de Impacto (JCR 2016): 7,671.
    - Áreas de Conocimiento:
        - Computer Science, Artificial Intelligence. Ranking 4/133 (Q1).
        - Engineering, Electrical & Electronic. Ranking 9/262 (Q1).

CFM-BD: a distributed rule induction algorithm for building Compact Fuzzy Models in Big Data classification problems

Mikel Elkano, Mikel Galar, Jose Sanz, Edurne Barrenechea, and Humberto Bustince, *Senior Member, IEEE*

***Abstract*—Interpretability has always been a major concern for fuzzy rule-based classifiers. The usage of human-readable models allows them to explain the reasoning behind their predictions and decisions. However, when it comes to Big Data classification problems, fuzzy rule-based classifiers have not been able to maintain the good trade-off between accuracy and interpretability that has characterized these techniques in non-Big Data environments. The most accurate methods build too complex models composed of a large number of rules and fuzzy sets, while those approaches focusing on interpretability do not provide state-of-the-art discrimination capabilities. In this paper, we propose a new distributed learning algorithm to construct accurate and compact Fuzzy Rule-Based Classification Systems for Big Data (CFM-BD). This method has been specifically designed from scratch for Big Data problems and does not adapt or extend any existing algorithm. The proposed learning process consists of three stages: 1) pre-processing based on the probability integral transform theorem; 2) rule induction inspired by CHI-BD and Apriori algorithms; 3) rule selection by means of a global evolutionary optimization. We conducted a complete empirical study to test the performance of our approach in terms of accuracy, complexity, and runtime. The results obtained were compared and contrasted with four state-of-the-art fuzzy classifiers for Big Data (FBDT, FMDT, Chi-Spark-RS, and CHI-BD). According to this study, CFM-BD is able to provide competitive discrimination capabilities using significantly simpler models composed of a few rules of less than 3 antecedents, employing 5 linguistic labels for all variables.**

***Index Terms*—Fuzzy Rule-Based Classification Systems, Evolutionary Algorithms, Big Data, Apache Spark, Probability Integral Transform, Quantile Function.**

I. Introduction

Fuzzy Rule-Based Classification Systems (FRBCSs) are powerful machine learning algorithms that provide interpretable and accurate models described by human-readable linguistic labels [1]. The main feature that makes FRBCSs stand out from other types of solutions is the ability to explain how outputs are inferred from inputs. Due to this valuable reasoning, these systems have been widely used in applications such as bioinformatics [2], medical problems [3], software fault prediction [4], anomaly intrusion detection [5], financial applications [6], image processing [7], and traffic congestion prediction [8], among others.

This work has been supported by the Spanish Ministry of Science and Technology under the project TIN2016-77356-P.

Mikel Elkano, Mikel Galar, Jose Sanz, Edurne Barrenechea, and Humberto Bustince are with the Departamento de Automática y Computación, Universidad Pública de Navarra, Navarra, 31006 Spain (e-mail: {mikel.elkano, mikel.galar, joseantonio.sanz, edurne.barrenechea, bustince}@unavarra.es)

However, the construction of interpretable models usually involves computationally intensive learning algorithms that require long runtimes. As a consequence, state-of-the-art FRBCSs have serious difficulties dealing with large-scale datasets. Given the increasing amount of information in the Digital Age, which is doubling every two years according to the study carried out by International Data Corporation (IDC) [9], the design of new scalable solutions presents many challenges. Some of the latest Big Data classification problems, such as HIGGS or HEPMASS[1], do not fit into the main memory of standard computers. Therefore, state-of-the-art sequential learning algorithms are not able to handle the whole training set on a single computer. Moreover, even if such quantity of data could be stored in an 8-GB RAM memory, the training process would probably lead to unacceptable runtimes.

In order to overcome these challenges, many researchers started to adapt well-known machine learning techniques to *distributed computing* paradigms such as MapReduce [10]–[12]. This methodology rapidly became very popular as a result of the development of open-source frameworks such as Apache Hadoop[2] and Apache Spark[3]. In the last few years, a number of distributed FRBCs based on either Hadoop or Spark have been proposed [13]–[22]. Although great progress has been made, most contributions do not achieve state-of-the-art results in terms of both accuracy and interpretability. Some of them perform several local optimization or learning processes to obtain an approximate global solution [15], [16], [19], [20], missing important patterns that could only be extracted when the training set is treated as a whole. Other methods produce too complex models that affect interpretability, mainly due to a large number of rules or fuzzy sets [14], [21], [22]. There are also other contributions that, from our point of view, do not include enough Big Data problems to assess their quality in the corresponding study [13], [18].

In this work, we propose a new distributed FRBCS to build compact and accurate models for Big Data classification problems. Our objective is to generate rule bases containing a few (short) rules using a small fixed number of fuzzy sets per variable, while achieving state-of-the-art classification performance. The proposed algorithm consists of three sequential stages:

[1] http://archive.ics.uci.edu/ml/datasets.html
[2] http://hadoop.apache.org
[3] https://spark.apache.org

1) *Pre-processing and partitioning*. Training data is transformed into a uniform distribution by applying the *probability integral transform theorem* [23], [24]. Next, the fuzzy sets are uniformly distributed in the new transformed space. In this manner, the partitions fit the actual distribution of the training data and can be safely recovered in the original space by making use of the *inverse distribution function* or *quantile function* [25].
2) *Rule induction process*. Rules are constructed by a novel learning algorithm inspired by CHI-BD [14] and Apriori [26] algorithms. First, the most frequent itemsets are extracted from the initial rules generated by CHI-BD and a pruning process is then carried out. Next, the itemsets are converted into candidate rules and a filtering and pruning process is performed to select the rules with the greatest discrimination capability.
3) *Evolutionary rule selection*. We implement our own distributed version of the CHC evolutionary algorithm [27] to optimize the rule base by means of rule selection. To the best of our knowledge, this is the first distributed solution for global evolutionary rule selection.

We must remark that all the stages process the whole dataset in a distributed fashion and the model obtained is not affected by the distribution of the partitions and the degree of parallelism. The full source code was written in Scala[4] 2.11 on top of Apache Spark 2.0.2 and is publicly available at GitHub[5] under the GPL license.

In order to assess the performance of our method, we carried out an empirical study using 6 Big Data classification problems available at UCI[6] and OpenML[7] repositories. Accuracy, complexity, and runtimes were analyzed and compared with the results obtained by three state-of-the-art fuzzy classifiers, i.e., CHI-BD [14], Chi-Spark-RS [16], and FMDT/FBDT [21]. Additionally, the scalability of our approach was measured with three well-known metrics used to evaluate distributed systems, i.e., *speedup*, *sizeup*, and *scaleup* [28], [29]. The experimental results show that CFM-BD is able to deal with large-scale datasets and achieve competitive accuracy rates while providing simpler models than the aforementioned algorithms.

This paper is organized as follows. Section II includes the basics of FRBCSs and Apache Hadoop/Spark and presents some related work. In Section III, we introduce the proposed distributed FRBCS for Big Data classification problems. The experimental framework is described in Section IV and the analysis of the empirical results is presented in Section V. Finally, Section VI concludes this paper.

II. Preliminaries

In this section we briefly describe some basic concepts and frameworks that are directly related to our proposal. First, we explain the basics of Fuzzy Rule-Based Classification Systems (Section II-A). Next, we introduce two popular frameworks used to handle Big Data environments called Apache Hadoop and Apache Spark (Section II-B). Finally, we present and discuss some recent related work (Section II-C).

A. Fuzzy Rule-Based Classification Systems

Fuzzy Rule-Based Classification Systems (FRBCSs) are well-known models that achieve a good trade-off between classification accuracy and interpretability. These systems provide an interpretable rule base containing human-readable rules composed of linguistic labels [1].

The two main components of FRBCSs are described hereafter.

1) *Knowledge base (KB)*: It is composed of both the rule base (RB) and the database (DB), where the rules and membership functions used to model the linguistic labels are stored, respectively.
2) *Fuzzy Reasoning Method (FRM)*: This is the mechanism used to classify examples employing the information stored in the KB.

In order to generate the KB, a fuzzy rule learning algorithm is applied using a training set TR composed of N labeled examples $x_i = (x_{i1}, \ldots, x_{iF})$ with $i \in \{1, \ldots, N\}$, where x_{if} is the value of the f-th feature ($f \in \{1, 2, \ldots, F\}$) of the i-th training example. Each example belongs to a class $y_i \in \mathbb{C} = \{C_1, C_2, ..., C_M\}$, where M is the number of classes in the problem.

The RB is composed of a set of rules having the following structure:

$$\begin{aligned} \text{Rule } R_j : \text{If } x_1 \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_F \text{ is } A_{jF} \\ \text{then Class = } C_j \text{ with } RW_j \end{aligned} \quad (1)$$

where R_j is the label of the j-th rule, $x = (x_1, \ldots, x_F)$ is an F-dimensional pattern vector that represents the example, A_{jf} is a linguistic label modeled by a triangular membership function, C_j is the class label and RW_j is the rule weight. In some cases rules might contain *don't care* linguistic labels that are ignored by the classifier. These labels can simply be removed from the RB, leading to variable rule lengths. Regarding the rule weight computation, in this work we use an adaptation of the well-known Penalized Certainty Factor (PFC) method [30] called Penalized Cost-Sensitive Certainty Factor (PCF-CS). This method minimizes the impact of the frequency of each class on the learning process by applying the following formula:

$$RW_j = \frac{matchClass - matchNotClass}{matchClass + matchNotClass}, \quad (2)$$

where

$$matchClass = \sum_{x_i \in ClassC_j} \mu_{A_j}(x_i) \cdot cost(y_i)$$

$$matchNotClass = \sum_{x_i \notin ClassC_j} \mu_{A_j}(x_i) \cdot cost(y_i).$$

$cost(y_i)$ is the misclassification cost associated with the class y_i and $\mu_{A_j}(x_i)$ is the matching degree between the example x_i and the antecedent part of the rule R_j. Although class costs were originally considered for binary classification problems,

[4]https://www.scala-lang.org
[5]https://github.com/melkano/cfm-bd
[6]http://archive.ics.uci.edu/ml/datasets.html
[7]https://www.openml.org/search?type=data

we have adapted their computation to multi-class problems as follows:

$$cost(y_i) = \frac{\max\limits_{m=1,\dots,M}(count(y_m))}{count(y_i)}, \quad (3)$$

where $count(y_i)$ is the number of examples belonging to the class y_i. As for the matching degree $\mu_{A_j}(x_i)$, it is defined as:

$$\mu_{A_j}(x_i) = \prod_{f=1}^{F} \mu_{A_{jf}}(x_{if}), \quad (4)$$

where $\mu_{A_{jf}}(x_{if})$ is the membership degree of the value x_{if} to the fuzzy set A_{jf} of the rule R_j. If A_{jf} is marked as *don't care*, the membership degree is set to 1.

In order to classify a new example x_i, the classifier runs an FRM composed of the following steps.

1) *Matching degree*. The strength of activation of the antecedent part of all rules in the rule base for the example x_i is computed (Eq. (4)).
2) *Association degree*. The association degree of the example x_i with each rule in the rule base is computed.

$$b_j(x_i) = \mu_{A_j}(x_i) \cdot RW_j \quad (5)$$

3) *Classification*. The final prediction is made based on the association degrees. In this work we use the *winning rule* [31] method, which predicts the class of the rule with the highest association degree:

$$class = arg \max_{m=1,\dots,M} \left(\max_{R_j \in RB;\, C_j = m} b_j(x_i) \right) \quad (6)$$

B. Apache Hadoop and Apache Spark

In the last few years, distributed computing has become very popular in the machine learning community thanks to open-source frameworks such as Apache Hadoop[8] and Apache Spark[9]. These frameworks provide a transparent distributed system that allows the user to focus only on data processing. The core of Hadoop consists of a distributed file system based on the Google File System [32] called *Hadoop Distributed File System* (HDFS) (storage layer) and an implementation of the MapReduce paradigm [10] (processing layer).

Spark was introduced as a generalization and an extension of the MapReduce paradigm. It has seemingly unseated Hadoop thanks to the so-called Directed Acyclic Graphs (DAGs) and the in-memory computing feature, which minimize the latency of multi-stage data flows that require multiple MapReduce jobs. Spark is built around the concept of *Resilient Distributed Datasets* (RDDs) [33], which represent distributed immutable data (partitioned data) and lazily evaluated operations (*transformations*). The execution of a user-defined algorithm consists of a sequence of *stages* composed of a number of transformations that are split into *tasks*. One stage consists only of transformations that do not require any shuffling/repartitioning process (e.g., *map* and *filter* operations). Tasks are executed by the so-called *executors*, which represent independent processes in the Java Virtual Machine (JVM) of a *worker* node. Finally, the result of all transformations is obtained by calling an *action* that computes and returns the result to the *driver* node. This data flow (Fig. 1) allows the user to run an indefinite number of MapReduce jobs within the same main program, supporting a much wider variety of algorithms and methods than Hadoop.

[8]http://hadoop.apache.org
[9]https://spark.apache.org

C. Related work

To the best of our knowledge, barely a dozen interpretable fuzzy methods have been proposed to deal with Big Data classification problems [13]–[22], [34], [35], to a large extent due to the excessive computational cost inherited from fuzzy learning algorithms. Given the success of fuzzy classifiers in a wide range of fields [2]–[8], designing scalable solutions seems worth the effort. In [36], the authors provide an overview of the progress and opportunities of fuzzy logic in Big Data environments.

In general, the aforementioned solutions are based on either incremental learning [34], [35] or distributed computing [13]–[22]. In the former case, training data is divided into several subsets called *episodes* that sequentially feed a classifier that incrementally learns from input data, including the knowledge acquired in previous episodes. The advantage of this approach is that it does not require a computing cluster to run the algorithm, since each episode is equivalent to a small-data classification problem in terms of computational cost. Regarding distributed solutions, the learning process is carried out by distributing training data across several computing nodes that perform partial computations in order to get the final model. The main difference between this strategy and incremental learning is that the former runs a single learning process in parallel using the whole training set, while the latter carries out several learning stages in a sequential fashion. Therefore, it is clear that the drawback of distributed approaches is the need for several computing nodes. However, incremental learning might miss important knowledge coming from inter-relations among data from different episodes, since it lacks the overview of the whole training set. In this work, we focus on the design of distributed methods.

Distributed learning algorithms might, in turn, tackle classification problems either by decomposing the original training data into several local sub-problems [15], [16], [19] or by performing a global distributed learning process [14], [17], [18], [21], [22], or even by combining these two approaches [13], [20]. In the former case, an independent local model is concurrently built in each subset (*chunk*) of data, so that the final classifier is obtained by aggregating all these models. In this manner, one could apply a well-known non-distributed algorithm to train each local model. However, similarly to incremental learning, the learning process becomes strongly dependent on the distribution of subsets and might miss important information available only when training data is treated as a whole. Regarding global distributed learning algorithms, the difficulty of parallelizing the training phase across several computing units is the main drawback.

Different strategies have been applied to obtain human-readable fuzzy models in Big Data classification problems,

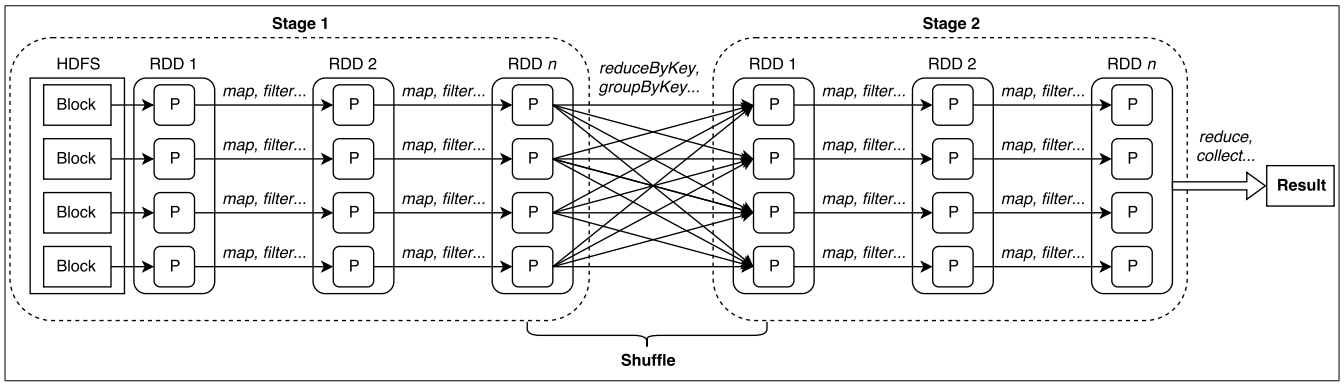


Fig. 1: Spark's data flow (P = Partition).

including fuzzy versions of decision trees (FDTs) [17], [21], sub-group discovery (SD) [20], associative classifiers (FACs) [13], [22], emerging patterns mining (EPM) [18], and rule-based classifiers (FRBCs) [14]–[17], [19]. In [17], a distributed version of C4.5 is used to extract a candidate rule base that is optimized by an evolutionary algorithm. Segatori et al. proposed a distributed FDT that exploits the classical Decision Tree implementation in Spark MLlib[10], extending the learning scheme by employing fuzzy information gain based on fuzzy entropy [21]. A new algorithm for SD called MEFASD-BD was presented in [20], which makes use of an evolutionary fuzzy system to extract fuzzy rules describing subgroups for each partition, though the quality of each solution is measured on the whole training set. Fuzzy logic was also used for EPM in Big Data by García-Vico et al. [18], applying a global evolutionary fuzzy system that employs the entire training set. Finally, different distributed versions of both FACs and FRBCs were proposed in [13]–[16], [19], [22].

However, the above-mentioned algorithms sacrifice either interpretability for classification accuracy or viceversa. Some algorithms focus on the accuracy and tend to generate too complex models having large amounts of rules [14], [15], [19], excessive rule lengths [14]–[16], [19], or a high number of fuzzy sets (linguistic labels) [21], [22]. On the other hand, those algorithms that optimize the interpretability of the model are not able to achieve state-of-the-art results in terms of accuracy [17]. There are also other contributions that, from our point of view, do not consider enough datasets to assess these aspects in Big Data environments [13], [18], [20].

III. CFM-BD: A NEW DISTRIBUTED FUZZY RULE INDUCTION ALGORITHM FOR BIG DATA

In this work we present CFM-BD, a new distributed fuzzy rule induction algorithm specifically designed for Big Data classification problems. The motivation behind our proposal is to build compact fuzzy models achieving state-of-the-art results in terms of both classification performance and interpretability.

In order to overcome this challenge, we propose a learning algorithm composed of three stages:

1) *Pre-processing and partitioning*. An adaptive fuzzy partitioning process is carried out by applying a novel pre-processing method based on the *probability integral transform theorem* [23], [24].
2) *Rule induction process*. Rules are constructed by a novel learning algorithm inspired by CHI-BD [14] and Apriori [26] algorithms.
3) *Evolutionary rule selection*. A rule selection process is conducted by a new distributed version of the CHC evolutionary algorithm [27].

We must remark that all stages are conducted by distributed processes that employ the whole training set. Therefore, no approximations are introduced throughout the whole execution of the algorithm. This feature allows the user to obtain exactly the same model regardless of the distribution of data partitions and the parallelization degree used for the execution.

The full source code was written in Scala[11] 2.11 on top of Apache Spark 2.0.2 and is publicly available at GitHub[12] under the GPL license.

A. Pre-processing and partitioning

The goal of this stage is to build fuzzy sets (linguistic labels) that fit the real distribution of the training data while keeping the number of fuzzy sets per variable constant (e. g., *low*, *medium*, *high*). This process is divided into two parts:

- Pre-processing: the original distribution of the training data is transformed into a uniform distribution. This transformation applies the *probability integral transform theorem* [23], [24], described in Theorem 1. This theorem implies that any dataset can be transformed into a new dataset where all the variables follow a uniform distribution, regardless of the original distribution.

  Theorem 1. *If X is a continuous random variable with cumulative distribution function (CDF) $F_X(x)$ and if $Y = F_X(X)$, then Y is a uniform random variable on the interval [0,1].*

[10] http://spark.apache.org/mllib

[11] https://www.scala-lang.org

[12] https://github.com/melkano/cfm-bd

Proof. Suppose that $Y = g(X)$ is a function of X where g is differentiable and strictly increasing. Thus, its inverse g^{-1} uniquely exists. The CDF of Y can be derived using

$$F_Y(y) = Prob\left(Y \leq y\right) = Prob\left(X \leq g^{-1}(y)\right) = F_X\left(g^{-1}(y)\right)$$

and its density is given by

$$f_Y(y) = \frac{d}{dy}F_Y(y) = \frac{d}{dy}F_X(g^{-1}(y)) = f_X(g^{-1}(y)) \cdot \frac{d}{dy}g^{-1}(y).$$

This procedure is called the CDF technique and allows the distribution of Y to be derived as follows:

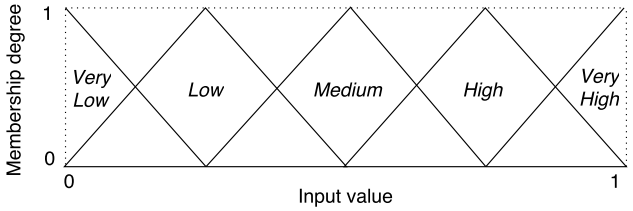
$$F_Y(y) = Prob\left(Y \leq y\right) = Prob\left(X \leq F_X^{-1}(y)\right) = F_X\left(F_X^{-1}(y)\right) = y$$

□

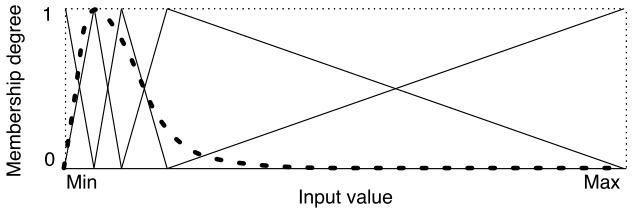
However, since the original distribution of the training set is unknown, we cannot compute the exact CDF. Instead, we propose computing the q-quantiles of the training set and compute an approximate CDF. To this end, for each variable, all the values are sorted and each quantile is extracted. If q is smaller than the number of examples in the training set, the CDF of a certain value is linearly interpolated on the interval $[Q_{i-1}, Q_i]$, Q_i being the first quantile greater than the value. If the value is smaller than the first quantile (Q_1) or greater than the last quantile (Q_{q-1}), the CDF is 0 or 1, respectively. Of course, the transformation of the testing set is performed by interpolating the CDF using the quantiles extracted from the training set.

- Partitioning: the fuzzy sets are built using triangular membership functions and uniformly distributed across the interval [0,1] in the new transformed space. It is worth noting that the definition of every single fuzzy set in the original space can be recovered by applying the *inverse distribution function* or *quantile function* [25]. In this case, for every point defining the triangular membership function, we would perform a linear interpolation between the two closest quantiles by computing the inverse of the linear function used to compute the CDF.

Fig. 2 shows an illustrative example of how fuzzy sets are distributed in the original and the transformed spaces of the variable M_Delta_R of SUSY. The dashed line represents the original distribution of the variable.



(a) Transformed space



(b) Original space (dashed line = original distribution of the variable)

Fig. 2: Fuzzy sets built for the variable M_Delta_R of SUSY.

B. Rule induction process

After the training data has been pre-processed and the fuzzy sets have been built, the rule base is constructed by applying a new rule induction algorithm specifically designed for Big Data. This process consists of two sequential stages that are inspired by some of the concepts introduced in CHI-BD [14] and Apriori [26] algorithms.

1) Search for the most promising itemsets: In this work, we consider all fuzzy sets (linguistic labels) and nominal values as *items*. When some of these items (one or more) appear together in a given transaction (example), they form an *itemset*. In order to find out the most promising itemsets in terms of discrimination capability, a procedure composed of three steps is applied.

1) Discretization of the examples: all the itemsets that are present in the training set are extracted by making use of the rule generation process performed by the CHI-BD algorithm [14]. This procedure consists in discretizing all the examples by computing the membership degree of each value to all the fuzzy sets of the corresponding variable. That is, each value is replaced with the fuzzy set leading to the highest membership degree. In case of nominal values, no discretization is conducted. Example 1 illustrates a discretized example, where the subscript indicates the variable that the fuzzy set corresponds to.

Example 1. Given the following example:

$0.15, 0.82, 0.51,$

it could be descretized as:

$Low_1, High_2, Medium_3$

In this manner, an n-dimensional example is transformed into an itemset of n items. After discretizing the example, all the possible subsets of items contained in the itemset are generated. The maximum length of these subsets is set by the user and determines the maximum length of the rules that will be built in subsequent stages. In this work, we have set this value to 3. Example 2 shows the itemsets extracted from Example 1.

Example 2. Given the following discretized example:

$Low_1, High_2, Medium_3,$

all the possible itemsets are:

$\{Low_1\}, \{High_2\}, \{Medium_3\}$,
$\{Low_1, High_2\}, \{Low_1, Medium_3\}, \{High_2, Medium_3\}$,
$\{Low_1, High_2, Medium_3\}$

2) Search for frequent itemsets: the *support* of each itemset is computed and only those having a minimum support are kept. In the original Apriori algorithm [26], the support of an itemset is the number of times that the itemset appears in the training set. In this work, the support of an itemset I is redefined as

$$supp_{crisp}(I) = \frac{count(I)}{N}, \quad (7)$$

where $count(I)$ is the original support used in [26] and N is the number of training examples. We have called it $supp_{crisp}$ to differentiate the crisp support used in this stage from the fuzzy support used in the fuzzy rules generation (Eq. (10) in Section III-B). Similarly to Apriori, an itemset is considered as *frequent* if its support equals or exceeds the *support threshold* set by the user. However, in our proposal this threshold depends on the cardinality of the itemset and the number of classes instead of being a fixed number:

$$minSupp_{crisp} = \frac{0.025}{|I| \cdot M}, \quad (8)$$

where $|I|$ is the number of items contained in the itemset and M is the number of classes in the problem. Unlike Apriori, in this work an itemset might be considered as frequent even if it contains non-frequent itemsets. In this manner, only those itemsets having a support lower than the support threshold are discarded. This difference comes from the fact that the minimum support of each itemset depends on the length of the itemset, so that small non-frequent itemsets are penalized more than larger ones. The reason for this adaptation is that the difference between the crisp and the fuzzy supports of larger itemsets is greater than that of smaller ones. As a consequence, valuable itemsets might be removed if this difference is not minimized.

3) Selection of the most confident itemsets: among the frequent itemsets, another filtering process is carried out based on the *confidence* of the itemsets. In this work, the confidence of an itemset is defined as:

$$conf_{crisp}(I) = \frac{\max\limits_{m=1,\ldots,M}(countClass(I, y_m))}{count(I)}, \quad (9)$$

where $countClass(I, y_m)$ counts the number of examples belonging to the class y_m in which the itemset I is present. Similarly to $supp_{crisp}$, we have called it $conf_{crisp}$ to differentiate this confidence from that used in the fuzzy rules generation. When the confidence is computed, some of the itemsets are discarded based on the following criteria:

- For each class, if 50% of the itemsets associated with the class have a confidence lower than a *confidence threshold* (in this work called $minConf_{crisp}$), only the top half is kept. Otherwise, those itemsets having a confidence greater than the confidence threshold are kept.
- If there exists a subset of the itemset that is more confident than the itemset itself and fulfills the previous criterion, the itemset is discarded. This means that large itemsets are kept only and only if they provide more discrimination capability than smaller ones.

We must remark that any occurrence of a certain itemset is always weighted by the cost associated with the class of the example in which the itemset is present (Eq. (3)). That is, both the support and the confidence count the occurrences by summing up the cost of the class of each example (same for N in Eq. (7)).

2) Construction of fuzzy rules: Based on the most promising frequent itemsets extracted in the previous stage, a rule base is created as follows:

1) Conversion from itemsets to candidate rules: every single itemset is transformed into one or more candidate rules. To this end, for a given itemset, the algorithm keeps track of the examples in which the itemset is present and it obtains their class labels. Then, a new candidate rule is generated for each of these classes. Example 3 illustrates this conversion.

Example 3. Given the following itemsets that have passed the previous filtering phase:

$\{High_2\}, \{Low_1, High_2\}, \{Low_1, High_2, Medium_3\}$.

And given the examples that have generated those itemsets:

$Low_1, High_2, Medium_3 \rightarrow C_1$
$Low_1, High_2, Medium_3 \rightarrow C_2$

We can extract the classes the itemsets belong to (C_1 and C_2) and generate the corresponding candidate rules:

IF A_2 is $High$ THEN C_1
IF A_2 is $High$ THEN C_2
IF A_1 is Low and A_2 is $High$ THEN C_1
IF A_1 is Low and A_2 is $High$ THEN C_2
IF A_1 is Low and A_2 is $High$ and A_3 is $Medium$ THEN C_1
IF A_1 is Low and A_2 is $High$ and A_3 is $Medium$ THEN C_2.

As we can observe in Example 3, two examples belonging to different classes might produce the same itemsets, and thus the algorithm would generate rules that share the antecedent part and have different consequent. This situation is known as a *conflict* and is resolved in the next step.

2) Computation of rule weights and conflict resolution: for each rule, the matching degree between the rule and all the examples in the training set is computed in order to obtain the rule weight, as shown in Eq. (2). This computation is performed in a distributed fashion by broadcasting all candidate rules across the worker nodes. In this manner, each worker computes only the partial sum of the matching degrees corresponding to the assigned partition. We must point out that this process represents a small fraction of the total computing time of the learning algorithm, since the number of rules considered at this point is generally low. When rule

weights are computed, conflicts are resolved by keeping the rule with the largest weight.

3) Filtering and pruning: first, the support and the confidence of every rule is computed according to Eq. (10) and (11), respectively, reusing the previously computed matching degrees:

$$supp_{fuzzy}(R) = \frac{matchClass + matchNotClass}{N} \tag{10}$$

$$conf_{fuzzy}(R) = \frac{matchClass}{matchClass + matchNotClass}, \tag{11}$$

using the $matchClass$, $matchNotClass$, and N defined in Eq. (2) and (7), respectively. Next, those rules having a support or a confidence lower than the support and the confidence thresholds are removed. In this work, the confidence threshold ($minConf_{fuzzy}$) has been set to 0.6. The support threshold is defined as:

$$minSupp_{fuzzy}(R) = \frac{0.05}{len(R) \cdot M}, \tag{12}$$

where $len(R)$ and M are the rule length and the number of classes in the problem, respectively. The usage of the rule length in this computation minimizes the penalizing effect that the product operation has in Eq. (10) when the number of antecedents increases. When the rules have been filtered based on their support and confidence, only the most confident rules of each class are kept. To this end, all the rules are grouped by class and length and sorted by confidence. Next, for each class C_m and rule length len_i with $m \in \{1, 2, ..., M\}$ and $i \in \{1, 2, ..., maxLen\}$, where $maxLen$ is the maximum rule length set by the user (in this work $maxLen = 3$), the first $NR_{C_j i}$ rules are taken:

$$NR_{C_j i} = L \cdot n \cdot prop_{len_i} \cdot \gamma, \tag{13}$$

where L is the number of fuzzy sets per variable, n is the number of variables of the problem, $prop_{len_i}$ is the proportion of rules with length i specified by the user, and γ is a hyperparameter that allows the user to set the priority between classification performance and model complexity. In this work we have set $prop_{len}$ = (0.2,0.3,0.5) to prioritize specific rules over general ones. As for γ (in this work $\gamma \in \{2, 4, 8\}$), high values cause the algorithm to build more rules and might enhance its classification performance. After filtering the most confident rules, a pruning process is carried out: those rules containing all the antecedents of a more confident and shorter rule are removed from the rule base. Example 4 illustrates this pruning process.

Example 4. Given the following rules:

R_1 : IF A_1 is Low and A_2 is $High$ THEN C_1
R_2 : IF A_1 is Low and A_2 is $High$ and A_3 is $Medium$ THEN C_1,

with $conf_{fuzzy}(R_1)$ = 0.83 and $conf_{fuzzy}(R_2)$ = 0.76, R_2 is discarded because R_1 is shorter, more confident, and all its antecedents are present in the antecedent part of R_2.

Fig. 3 and 4 show the pseudo-code of the rule induction algorithm and the four Spark stages launched during the process, respectively. We must highlight two aspects in Fig. 3:

- In line 17, candidate rules are grouped by the antecedent part, and thus conflicting rules fall into the same key-value pair. In this manner, *map* and *filter* transformations can compute the support, the confidence, and the weight of all conflicting rules at once.
- Functions *is_frequent ()* and *is_confident ()* check whether the support and the confidence of a given itemset/rule are greater than the corresponding thresholds, respectively.

As mentioned in Section II-B, the shuffling operation occurs only between stages, and thus transformations such as *map* and *filter* run in parallel without communication overhead.

Function generate_rule_base (TR)
Input: A pre-processed training set TR containing N labeled examples x_i.
Output: A rule base RB.

Begin

1: ***# 1. Search for the most promising itemsets***
2: ***# 1.1 Discretization of the examples***
3: $TR_d \leftarrow TR.map\ (x_i \leftarrow discretize\ (x_i))$
4: $Itemsets \leftarrow TR_d.map\ (x_i^d \leftarrow extract\_itemsets\ (x_i^d))$
5: ***# 1.2 Search for frequent itemsets***
6: $SuppConf \leftarrow Itemsets$ $.reduceByKey\ (itemset \leftarrow support\_and\_confidence\ (itemset))$
7: $Itemsets_{Freq} \leftarrow SuppConf.filter\ (is\_frequent\ (itemset))$
8: ***# 1.3 Selection of the most confident itemsets***
9: $Itemsets_{Conf} \leftarrow Itemsets_{Freq}.filter\ (is\_confident\ (itemset))$
10: $Itemsets_{Prom} \leftarrow distributed\_pruning\ (Itemsets_{Conf})$
11: ***# 2. Construction of fuzzy rules***
12: ***# 2.1 Conversion from itemsets to candidate rules***
13: $Rules_{cand} \leftarrow Itemsets_{Prom}.map\ (itemset \leftarrow rule\ (itemset))$
14: $Rules_{broad} \leftarrow broadcast\ (Rules_{cand}.collect\ ())$
15: ***# 2.2 Computation of rule weights and conflict resolution***
16: $matchings \leftarrow TR_d.map\ (x_i \leftarrow matching\ (Rules_{broad}))$
17: $SuppConfWght \leftarrow matchings$ $.reduceByKey\ (rule \leftarrow support\_confidence\_weight\ (rule))$
18: $Rules_{no\_conflicts} \leftarrow SuppConfWght$ $.map\ (rule \leftarrow resolve\_conflicts\ (rule))$
19: ***# 2.3 Filtering and pruning***
20: $Rules_{freq} \leftarrow Rules_{no\_conflicts}$ $.filter\ (rule \leftarrow is\_frequent\ (rule))$
21: $Rules_{conf} \leftarrow Rules_{freq}.filter\ (rule \leftarrow is\_confident\ (rule))$
22: $Rules \leftarrow distributed\_pruning\ (Rules_{conf})$
23: **RETURN** *build_rule_base* ($Rules$)

End

Fig. 3: Pseudo-code of the rule induction algorithm.

C. Evolutionary rule selection

When the rule base has been created, a rule selection process is carried out in order to obtain a compact and accurate model. To this end, we apply the CHC evolutionary algorithm (EA) [27] because of its ability to deal with complex search spaces [37] and the good results achieved by this EA in state-of-the-art FRBCSs, such as FARC-HD [38] or IVTURS [39]. Unlike other methods that make use of CHC [15], [16], we have implemented a new distributed version that performs a global optimization process by evaluating the quality (fitness) of each individual considering the whole training set.

Next, the main features of the CHC algorithm are described:

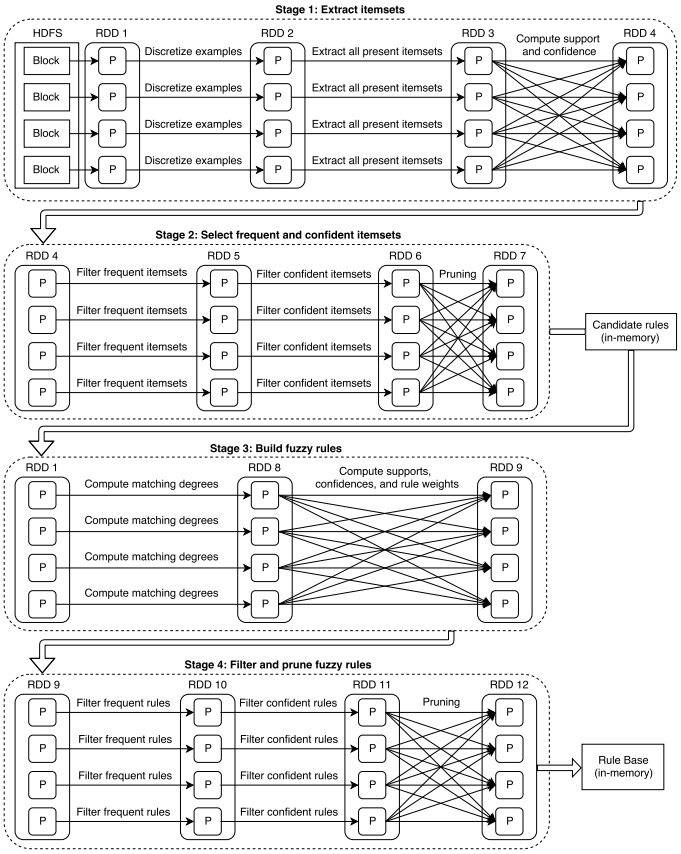


Fig. 4: Spark stages launched during the rule induction process (P = Partition).

- *Coding Scheme*. Each chromosome $C = (c_1, c_2, ..., c_{NR})$ is coded as a binary vector of NR elements, NR being the number of rules contained in the rule base. Each element is associated with a certain rule and determines whether the rule is selected or not. In this manner, $If\ c_i = 1\ Then\ (R_i \in RB)\ Else\ (R_i \notin RB)$, where RB is the final optimized rule base.
- *Initial Gene Pool*. To include the initial rule base as a candidate solution, the initial pool is obtained with the first individual setting all genes to 1 and the remaining individuals being generated at random.
- *Chromosome Evaluation*. Since the goal of our method is to build a compact and accurate model, both the accuracy and the complexity of the rule base need to be considered. To this end, the quality or the *fitness* of a chromosome is determined by the same equation used in FARC-HD:

$$Fitness(C) = acc - \delta \cdot \frac{NR_{initial}}{NR_{initial} - NR + 1}, \quad (14)$$

where $NR_{initial}$ and NR are the number of rules in the initial and the current rule bases, respectively, and acc is the accuracy obtained by the current model. In order to maintain the accuracy for all classes, we use one of the most common metrics for imbalanced datasets, that is, the geometric mean (GM) [40]:

$$acc = GM = \sqrt[M]{\frac{\#Hits_{C_1}}{N_{C_1}} \cdot \frac{\#Hits_{C_2}}{N_{C_2}} \cdot ... \cdot \frac{\#Hits_{C_M}}{N_{C_M}}}, \quad (15)$$

where $\#Hits_{C_m}/N_{C_m}$ is the percentage of correctly classified examples belonging to class C_m. Since the evaluation of the whole population is the most computationally expensive task in EAs, we have implemented our own distributed version of CHC to parallelize this computation across worker nodes. When computing the fitness of a whole population, all the individuals are sent to every single worker. Then, each worker computes a *partial confusion matrix* for each individual considering only the examples contained in its partitions. Finally, these partial matrices are summed up and the exact accuracy is obtained. In this manner, the fitness of an individual is computed using the whole training set without introducing any approximation. Additionally, in order to avoid repeated computations, we store an RDD containing the pre-computed association degrees of each rule in the initial rule base for all the examples. This allows CHC to use this RDD as a look-up-table when classifying a certain partition. We must point out that the look-up-table is distributed across worker nodes, and thus only the partition assigned to the worker is loaded in the main memory.

- *Crossover Operator*. The half uniform crossover scheme (HUX) is applied [41]. The HUX crossover randomly interchanges half of the genes that are different in the parents, ensuring the maximum distance between the offspring and their parents (exploration).
- *Restarting Approach*. The usage of a restarting approach helps the EA avoid local optima. When the population is restarted, the best chromosome is kept and the remaining are generated at random by keeping a certain percentage (Γ) of the genes contained in the best chromosome (set by the user).

CHC uses an incest prevention mechanism when applying the crossover operator: two parents are crossed only if their hamming distance divided by 2 is greater than a given threshold D. This threshold is initialized as the maximum possible distance between two individuals (number of genes in a chromosome) divided by four. When no individuals are added to the next generation, the value of D is decreased by φ times its initial value, φ being set by the user (in this work 0.01). When D is below zero, and if the maximum number of restarts without improvement has not been reached ($maxRestarts$), the population is restarted.

IV. Experimental Framework

In this section, we present the framework used to develop the experiments carried out in Section V. Firstly, we describe the datasets selected for the experimental study (Section IV-A). Next, we show the parameters considered for each method (Section IV-C). Finally, we introduce the performance and scalability measures used to evaluate the methods (Section IV-B).

A. Datasets

In order to develop the experimental study, we considered 6 Big Data classification problems available at UCI[13] and OpenML[14] repositories. Table I shows the description of the datasets indicating the number of instances (#Instances), real (R)/integer(I)/categorical(C)/total(T) attributes (#Attributes), and classes (#Classes). The names of BNG Australian (BNG), Covertype (COV), HEPMASS (HEPM), and KDDCup1999 (KDD) have been shortened. All the experiments were carried out using a *5-fold stratified cross-validation scheme*. To this end, we randomly split the dataset into five partitions of data, each one containing 20% of the examples, and we employed a combination of four of them (80%) to train the system and the remaining one to test it. Therefore, the result of each dataset was computed as the average of the five partitions.

TABLE I: Description of the datasets.

Dataset	#Instances	#Attributes				#Classes
		R	I	C	T	
BNG	1,000,000	8	6	0	14	2
COV	581,012	10	0	44	54	7
HEPM	10,500,000	28	0	0	28	2
HIGGS	11,000,000	28	0	0	28	2
KDD	4,898,431	26	0	15	41	5
SUSY	5,000,000	18	0	0	18	2

B. Performance metrics and scalability measures

The classification accuracy of the different methods was measured using the accuracy rate (Acc), the average accuracy rate per class (Acc_{Class}), and the geometric mean (GM) [40], defined as follows:

$$Acc = \frac{\#Hits}{N}, \tag{16}$$

$$Acc_{Class} = \frac{\sum_{m=1}^{M} \frac{\#Hits_{C_m}}{N_{C_m}}}{M}, \tag{17}$$

$$GM = \sqrt[M]{\frac{\#Hits_{C_1}}{N_{C_1}} \cdot \frac{\#Hits_{C_2}}{N_{C_2}} \cdot \ldots \cdot \frac{\#Hits_{C_M}}{N_{C_M}}}, \tag{18}$$

where $\#Hits$ is the number of correctly classified examples and N is the total number of examples, respectively. $\#Hits_{C_m}$ and N_{C_m} consider only the examples belonging to class C_m. Since some of the datasets considered in the experiments are imbalanced, Acc_{Class} and GM provide more information about the actual discrimination capability than Acc.

Additionally, we measured the scalability of our approach by applying three well-known metrics used to evaluate distributed systems, i.e., speedup, sizeup, and scaleup [28], [29].

- *Speedup*: the data size is kept constant and the number of cores is increased. An ideal distributed algorithm should feature a linear speedup, that is, a system with m cores must provide a speedup of m. However, in practice a linear speedup is difficult to obtain due to communication and synchronization overhead.

$$Speedup(m) = \frac{\text{runtime on 1 core}}{\text{runtime on } m \text{ cores}} \tag{19}$$

- *Sizeup*: the number of cores is kept constant and the data size is increased. Sizeup measures how much longer it will take to process an m-times larger dataset.

$$Sizeup(data, m) = \frac{\text{runtime for processing } m \cdot data}{\text{runtime for processing } data} \tag{20}$$

- *Scaleup*: the ability of a system to run an m-times greater job with m-times larger system is measured, whose ideal value should be 1 (runtime of the baseline system).

$$Scaleup(data, m) = \frac{\text{runtime for processing } data \text{ on 1 core}}{\text{runtime for processing } m \cdot data \text{ on } m \text{ cores}} \tag{21}$$

C. Methods and parameters setup

We included all the open-source fuzzy classifiers available for Big Data so far, i.e., CHI-BD [14], Chi-Spark-RS [16], and FMDT/FBDT [21]. Chi-FRBCS-BigData [19] was not included in the comparisons because CHI-BD showed better performance in terms of accuracy and complexity in [14]. Table II shows the parameters considered for each method throughout the experiments. In all cases, we set the values suggested by the authors in the original papers.

TABLE II: Parameters used for each method.

Algorithm	*Parameters*

CFM-BD (rule induction)	#Fuzzy sets per variable = 5
	Inference = winning rule
	Rule weight = PCF-CS
	$maxLen$ = 3; $prop_{len}$ = (0.2, 0.3, 0.5)
	$minConf_{crisp}$ = 0.7; $minConf_{fuzzy}$ = 0.6
	$\gamma \in \{2, 4, 8\}$

CFM-BD (rule selection)	#Individuals = 50; #Evaluations = 10,000 $maxRestarts$ = 3; $D = NR_{initial}$ / 4 δ = 0.15; Γ = 0.35; φ = 0.01
FMDT	Impurity = entropy; T-norm = product $maxBins$ = 32; $maxDepth$ (β) = 5 γ = 0.1%; $\phi = 0.02 \cdot N$; $\lambda = 10^{-4} \cdot N$
FBDT	Impurity = entropy; T-norm = product $maxBins$ = 32 γ = 0.1%; ϕ = 1; λ = 1

CHI-BD (cost-sensitive)	#Fuzzy sets per variable = 3
	Inference = winning rule
	Rule weight = PCF-CS
	Number of rule subsets = 4
	Minimum #occurrences
	for frequent subsets = 10
	Maximum #rules per reducer = 400,000

Chi-Spark-RS (cost-sensitive)	#Fuzzy sets per variable = 3
	Inference = winning rule
	Rule weight = PCF-CS; T-norm = product
	#Individuals = 50; #Evaluations = 1,000
	α (for fitness) = 0.7

In the case of CFM-BD, there is an extra boolean parameter called $cost\_sensitive$ that enables/disables the cost-sensitive

[13] http://archive.ics.uci.edu/ml/datasets.html
[14] https://www.openml.org/search?type=data

mode. When it is off, the cost associated with each class ($cost(y_m)$) is set to 1, so that the learning algorithm does not take class frequencies into account. Also, the pruning of itemsets described in Section III-B is not carried out for each class. Instead, the top half of all itemsets is considered regardless of their class. Similarly, the most confident fuzzy rules are extracted without considering their class, so that rules are grouped only by length. Consequently, $NR_{C_j i}$ (Eq. (22)) is replaced with NR_i:

$$NR_i = L \cdot n \cdot prop_{len_i} \cdot \gamma \cdot M, \tag{22}$$

M being the number of classes. This way, the user can turn on/off the cost-sensitive mode of CFM-BD by simply setting $cost\_sensitive$ to ON/OFF. When optimizing Acc, the non cost-sensitive version ($cost\_sensitive$ OFF) performs better in general, while the best Acc_{Class} and GM are obtained when using the cost-sensitive version ($cost\_sensitive$ ON). The Acc reported in this work corresponds to the non cost-sensitive CFM-BD, while Acc_{Class} and GM correspond to the cost-sensitive version.

Additionally, we include two versions of CFM-BD: CFM-BD and CFM-BD^L. The former corresponds to the original method introduced in Section III, while the latter is a lightweight mode that omits the evolutionary rule selection process. Therefore, the only difference between these two versions is that CFM-BD^L gets rid of the third stage described in Section III-C. The original CFM-BD provides more accurate and compact models, whereas CFM-BD^L is meant to achieve a good trade-off among classification performance, model complexity, and execution time.

Regarding the cluster used for running the algorithms, it is composed of 6 slave nodes and a master node connected via 1Gb/s Ethernet LAN network. Half of slave nodes have 2 Intel Xeon E5-2620 v3 processors at 2.4 GHz (3.2 GHz with Turbo Boost) with 12 virtual cores in each one (where 6 of them are physical). The other half are equipped with 2 Intel Xeon E5-2620 v2 processors at 2.1 GHz with the same number of cores as the previous ones. The master node is composed of an Intel Xeon E5-2609 processor with 4 physical cores at 2.4 GHz. All slave nodes are equipped with 64 GB of RAM memory, while the master works with 32 GB of RAM memory. With respect to the storage specifications, all nodes use Hard Disk Drives featuring a read/write performance of 128 MB/s. The entire cluster runs on top of CentOS 6.5 + Apache Hadoop 2.6.0 + Apache Spark 2.0.2.

Except for FMDT/FBDT, the number of partitions/cores used for the execution of the algorithms equals the maximum number supported by our cluster, i.e., 128. In the case of FMDT/FBDT, we found that using more than 24 cores had a negative impact on runtimes when setting the configuration recommended by the authors. Thus, the number of cores used for FMDT/FBDT was 24. In all cases, we assigned 4 cores to every single executor in order to ensure full HDFS write throughput while minimizing memory replication overhead (e.g. broadcast variables).

V. Experimental study

In this section we describe the empirical study carried out to assess the performance of the proposed method (CFM-BD), which consists of two parts:

1) We tested CFM-BD in six Big Data classification problems and compared its performance with that provided by four state-of-the-art fuzzy classifiers (FMDT/FBDT [21], Chi-Spark-RS [16], and CHI-BD [14]) (Section V-A). More specifically, the performance of these methods was evaluated in terms of classification accuracy, model complexity, and runtimes.
2) We assessed the scalability of our approach with three well-known metrics used to evaluate distributed systems, i.e., *speedup*, *sizeup*, and *scaleup* [28], [29] (Section V-B).

A. Classification performance and complexity

Table III shows the classification performance of each method measured with Acc, Acc_{Class}, and GM. The second column indicates the γ used for CFM-BD (Eq. (13)) and the maximum depth considered for FBDT and FMDT (β). In order to replicate the configurations suggested in [21], the maximum depth used for FMDT is always 5. As we can observe, larger γ's do not imply better classification performance in the case of CFM-BD, while deeper trees generally provide more accurate models in FBDT. Before analyzing and comparing each method, we must mention that the implementation of Chi-Spark-RS available at GitHub[15] does not support multi-class problems, and thus we could not run this algorithm on KDD and COV. Besides, this method was not able to tackle HIGGS, HEPMASS, and SUSY within a period of 48 hours, and hence no results are given on these datasets for this method. Consequently, we have decided to ignore Chi-Spark-RS in the analysis of the results. Also, FMDT and CHI-BD ran out of memory on HEPMASS and we were not able to extract any result on this dataset.

Regarding Acc, FBDT and FMDT generally surpassed the rest of methods, including CFM-BD. In the case of CHI-BD, its performance is comparable to that of CFM-BD on BNG and KDD, but drastically drops on COV, HIGGS and SUSY. Although less accurate in general, CFM-BD achieves competitive results with respect to FBDT and FMDT. However, CFM-BD was not designed to maximize the Acc. Instead, our goal was to develop a FRBCS that is able to maintain its discrimination capability for all classes. We decided to focus on this aspect because, from our point of view, the actual discrimination capability of classifiers when facing multi-class problems should be measured considering the accuracy for all classes. For this reason, we have considered the Acc_{Class} and the GM measures in addition to Acc. Looking at Table III, we can observe that CFM-BD is able to provide competitive classification performance with respect to FBDT and FMDT in terms of Acc_{Class} and GM, outperforming CHI-BD in general. As for the different versions of CFM-BD, the fast

[15] https://github.com/aFdezHilario/Chi-Spark-RS

mode (CFM-BD^L) is about 1-2% less accurate than the original CFM-BD.

Besides classification performance, the objective of this work is to build compact and interpretable models. In order to measure the complexity of the different models, we have analyzed three aspects: number of rules, average rule length ($\overline{RL}$), and average number of fuzzy sets per variable ($\overline{FS}$). In the case of decision trees (FBDT and FMDT), leaves are equivalent to rules and their length is comparable to the depth of the tree. As we can observe in Table IV, CFM-BD provides much simpler rule bases than the rest of methods, even in the case of the non-evolutionary version (CFM-BD^L). In terms of the number of rules, only FBDT is able to achieve comparable results. However, the models of FMDT and FBDT employ a large number of fuzzy sets per variable that varies from one variable to another, while CFM-BD uses a fixed number (5) for all variables. Furthermore, the rules constructed by FBDT might contain duplicate linguistic labels in the antecedent part. As a result, the rule bases built by FMDT and FBDT are too complex to interpret. On the other hand, although CHI-BD uses exactly the same number of fuzzy sets per variable, it generates too many rules composed of a high number of antecedents.

The execution times of each method are shown in Table V. As expected, the original evolutionary version of CFM-BD is significantly slower than the rest. Although we prioritized the interpretability of the model over the execution time, we have proposed a non-evolutionary lightweight version of CFM-BD^L that is much faster than CFM-BD, as shown in Table V. Of course, this version sacrifices both accuracy and interpretability to reduce execution times.

Overall, CFM-BD achieves state-of-the-art discrimination capabilities with respect to the best performing algorithms (FBDT and FMDT), while providing much simpler models that can be interpreted. The models of CFM-BD generally consist of a few rules composed of less than 3 antecedents when keeping γ below 4, whereas other methods generate thousands of rules containing more than 3 antecedents. Additionally, the experimental results have revealed that the non-evolutionary rule induction algorithm of CFM-BD (CFM-BD^L) provides a good trade-off among classification performance, model complexity, and execution time. We must mention that the model and time complexities of CFM-BD shown in this paper correspond only to the cost-sensitive mode, since the complexities are similar regardless of this feature.

B. Scalability

Finally, we study the efficiency of our approach in terms of speedup, sizeup, and scaleup [28], [29] by testing CFM-BD on several reduced versions of HIGGS and varying the number of cores used for the execution. More specifically, we take 8 cores and 10% of HIGGS as the baseline case (m = 1) and we gradually double both the number of cores and the data size (maintaining the original class distribution), until 64 cores and 80% of HIGGS. This way, for each number of cores (8, 16, 32, 64) we run the model using 10%, 20%, 40%, and 80% of data. In addition to the total execution time, we test the

TABLE III: Classification performance of each method.

Dataset	γ; β	CFM-BD	CFM-BD^L	FBDT	FMDT	Chi-Spark-RS	CHI-BD
			Accuracy rate % (Acc)				
BNG	2; 5	86.46	85.35	78.83	80.23	75.19	84.21
	4; 10	86.48	85.22	80.17			
	8; 15	**86.61**	85.22	80.17			
COV	2; 5	72.54	69.68	69.31	**94.28**	-	51.69
	4; 10	72.43	69.75	76.30			
	8; 15	72.22	70.18	82.87			
HEPM	2; 5	90.63	89.17	90.61	-	-	-
	4; 10	90.83	89.25	91.15			
	8; 15	90.84	89.23	**91.40**			
HIGGS	2; 5	65.15	62.13	66.39	71.54	-	58.54
	4; 10	68.12	63.22	70.60			
	8; 15	68.43	63.47	**72.23**			
KDD	2; 5	99.07	98.80	99.88	**99.99**	-	99.65
	4; 10	98.81	98.80	**99.99**			
	8; 15	98.80	98.80	**99.99**			
SUSY	2; 5	77.41	75.86	77.31	79.29	-	64.89
	4; 10	78.34	76.11	79.09			
	8; 15	78.93	76.44	**79.70**			
Dataset	γ; β	CFM-BD	CFM-BD^L	FBDT	FMDT	Chi-Spark-RS	CHI-BD
			Average accuracy rate % per class (Acc_{Class})				
BNG	2; 5	86.24	85.06	77.44	78.96	75.35	85.02
	4; 10	86.33	85.02	78.81			
	8; 15	**86.41**	85.03	78.84			
COV	2; 5	61.86	58.80	44.59	**87.25**	-	66.73
	4; 10	65.90	61.62	62.17			
	8; 15	68.50	62.84	75.65			
HEPM	2; 5	90.67	89.17	90.61	-	-	-
	4; 10	90.66	89.25	91.15			
	8; 15	90.66	89.10	**91.40**			
HIGGS	2; 5	68.04	64.69	66.23	71.43	-	58.48
	4; 10	68.78	65.27	70.47			
	8; 15	68.89	65.25	**72.09**			
KDD	2; 5	96.41	95.16	59.47	92.15	-	83.94
	4; 10	96.01	92.95	89.84			
	8; 15	**97.61**	93.38	90.60			
SUSY	2; 5	76.39	74.51	76.57	78.59	-	62.42
	4; 10	77.05	74.86	78.35			
	8; 15	78.25	75.36	**79.01**			
Dataset	γ; β	CFM-BD	CFM-BD^L	FBDT	FMDT	Chi-Spark-RS	CHI-BD
			Geometric Mean (GM)				
BNG	2; 5	.8624	.8488	.7685	.7847	.7534	.8483
	4; 10	.8633	.8483	.7825			
	8; 15	**.8641**	.8484	.7832			
COV	2; 5	.5855	.5377	.2861	**.8680**	-	.6419
	4; 10	.6349	.5763	.5738			
	8; 15	.6668	.5907	.7417			
HEPM	2; 5	.9066	.8910	.9059	-	-	-
	4; 10	.9065	.8919	.9114			
	8; 15	.9065	.8907	**.9139**			
HIGGS	2; 5	.6800	.6456	.6619	.7140	-	.5847
	4; 10	.6873	.6526	.7043			
	8; 15	.6885	.6525	**.7205**			
KDD	2; 5	.9636	.9512	.0000	.9076	-	.7594
	4; 10	.9596	.9275	.8833			
	8; 15	**.9759**	.9317	.8880			
SUSY	2; 5	.7633	.7450	.7603	.7815	-	.5524
	4; 10	.7702	.7485	.7787			
	8; 15	.7816	.7535	**.7858**			

TABLE IV: Complexity of each method.

Dataset	γ; β	CFM-BD			CFM-BD^L			FBDT			FMDT			Chi-Spark-RS			CHI-BD		
		#rules	$\overline{RL}$	$\overline{FS}$	#rules	$\overline{RL}$	$\overline{FS}$	#rules	$\overline{RL}$	$\overline{FS}$	#rules	$\overline{RL}$	$\overline{FS}$	#rules	$\overline{RL}$	$\overline{FS}$	#rules	$\overline{RL}$	$\overline{FS}$
BNG	2; 5	5	2.00	5.00	239	2.51	5.00	32	5.00	6.04	83,044	3.02	6.04						
	4; 10	7	2.14	5.00	454	2.58	5.00	666	9.69	6.04				6,493	14.00	3.00	8,720	14.00	3.00
	8; 15	10	2.20	5.00	858	2.63	5.00	6,302	14.22	6.04									
COV	2; 5	164	2.70	5.00	2,226	2.79	5.00	31	4.97	18.84	268,677	3.39	18.84						
	4; 10	539	2.84	5.00	3,994	2.87	5.00	779	9.87	18.84				-	-	-	5,135	54.00	3.00
	8; 15	1,573	2.95	5.00	6,811	2.92	5.00	8,723	14.36	18.84									
HEPM	2; 5	7	1.00	5.00	462	2.49	5.00	30	4.93	22.20	-	-	-						
	4; 10	7	1.14	5.00	884	2.56	5.00	681	9.84	22.20				-	-	-	-	-	-
	8; 15	13	1.69	5.00	1,710	2.61	5.00	13,805	14.77	22.20									
HIGGS	2; 5	23	2.48	5.00	443	2.61	5.00	32	5.00	13.01	6,414,575	3.25	13.01						
	4; 10	31	2.45	5.00	882	2.62	5.00	849	9.89	13.01				-	-	-	666,068	28.00	3.00
	8; 15	49	2.61	5.00	1,471	2.71	5.00	17,390	14.79	13.01									
KDD	2; 5	60	2.37	5.00	1,619	2.58	5.00	24	4.83	15.93	8,042	3.18	15.93						
	4; 10	229	2.59	5.00	3,170	2.64	5.00	171	8.80	15.93				-	-	-	5,646	41.00	3.00
	8; 15	748	2.76	5.00	5,367	2.75	5.00	369	11.89	15.93									
SUSY	2; 5	13	1.00	5.00	324	2.44	5.00	32	5.00	22.60	5,225,134	3.68	22.60						
	4; 10	16	1.50	5.00	604	2.53	5.00	718	9.78	22.60				-	-	-	9,505	18.00	3.00
	8; 15	21	1.76	5.00	1,168	2.58	5.00	11,054	14.63	22.60									

TABLE V: Runtime (s) of each method.

Dataset	γ; β	CFM-BD	CFM-BD^L	FBDT	FMDT	Chi-Spark-RS	CHI-BD
BNG	2; 5	4,496	24	65	84		
	4; 10	4,299	24	84		1,740	77
	8; 15	6,721	23	108			
COV	2; 5	9,394	213	29	113		
	4; 10	12,395	217	49		-	62
	8; 15	26,458	220	87			
HEPM	2; 5	11,099	996	325	-		
	4; 10	24,655	1,011	391		-	-
	8; 15	60,697	1,003	623			
HIGGS	2; 5	23,253	1,252	309	5,238		
	4; 10	38,134	1,213	474		-	9,658
	8; 15	58,716	1,259	716			
KDD	2; 5	28,510	175	105	77		
	4; 10	53,927	176	157		-	81
	8; 15	126,568	169	186			
SUSY	2; 5	7,256	150	148	1,392		
	4; 10	12,373	151	235		-	103
	8; 15	18,726	158	364			

TABLE VI: Runtime (s) of CFM-BD on HIGGS.

Stage	Data size	8 cores	16 cores	32 cores	64 cores
Extraction of candidate rules	10%	1,486	754	397	269
	20%	2,549	1,495	811	470
	40%	6,084	2,687	1,537	949
	80%	12,226	5,850	3,169	1,788
Whole learning algorithm	10%	27,445	13,872	7,949	5,434
	20%	68,846	23,000	14,576	11,296
	40%	108,492	71,415	29,505	20,410
	80%	478,612	106,728	74,642	37,018

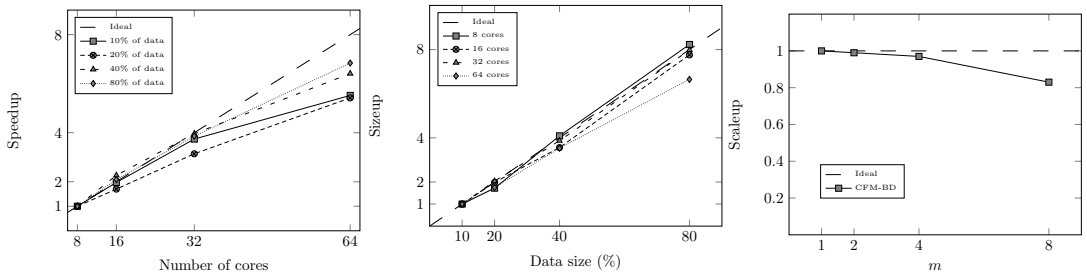
efficiency of the most critical stage of the learning algorithm: the extraction of candidate rules. With the runtimes obtained in these executions we build the matrices shown in Table VI, which are used to compute the speedup, sizeup, and scaleup (Figures 5 and 6). All the executions haven been carried out using $\gamma = 4$.

As we can observe in Fig. 5, the extraction of candidate rules shows almost linear speedup and sizeup and is able to maintain the scalability above 0.83. When it comes to the whole learning process, the plots shown in Fig. 6 reveal large variations among the different scenarios. The reason behind this behavior is the pre-computation step performed before the evolutionary optimization. Since the matching degrees of the rules do not change during the rule selection process, a distributed look-up table stores the matching degrees between the rules and the examples before launching the evolutionary algorithm. When this table does not fit into a cached RDD, the efficiency of the evolutionary optimization drastically drops. Except for these cases, Fig. 6 shows that the speedup and the sizeup of the whole learning process is almost linear and the scalability remains above 0.74.

VI. Conclusions

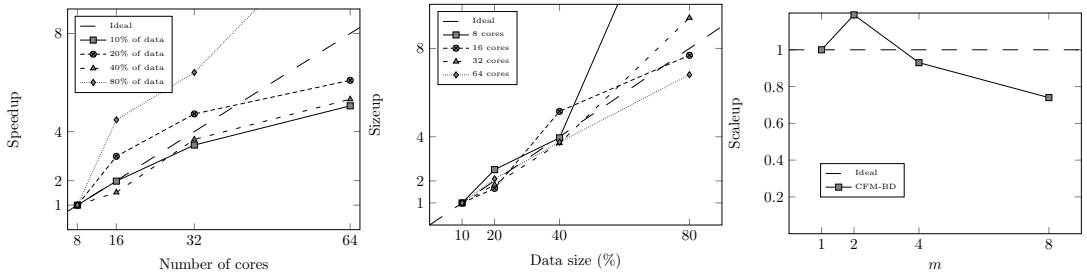
In this paper we have presented a new distributed FRBCS for Big Data classification problems (CFM-BD). The majority of fuzzy classifiers designed for Big Data so far are based on adaptations or extensions of existing learning algorithms. None of these approaches has been specifically designed from scratch to provide a good trade-off between accuracy and interpretability in Big Data problems.

The goal of this work was to build compact and interpretable models that achieve competitive classification performance. To this end, we have proposed a new rule induction process inspired by CHI-BD [14] and Apriori [26] algorithms called CFM-BD. Although it employs concepts introduced by these two methods, CFM-BD does not adapt, extend, or combine any of them. Instead, it applies a new learning algorithm composed of three stages: preprocessing, rule induction, and global evolutionary rule selection. All these stages have been specifically designed for Big Data from scratch in order to process the whole training set in a distributed fashion and



(a) Speedup (b) Sizeup (c) Scaleup

Fig. 5: Efficiency of the extraction of candidate rules on HIGGS.



(a) Speedup (b) Sizeup (c) Scaleup

Fig. 6: Efficiency of the whole learning algorithm on HIGGS.

perform global optimization tasks that do not introduce any approximation error. As a result, CFM-BD always provides exactly the same model regardless of the degree of parallelism used for the execution.

The experimental results show that the models generated by CFM-BD are significantly simpler than the rest. In terms of the number of rules, CFM-BD generally builds a few rules composed of less than 3 antecedents, while other methods generate thousands of rules containing more than 3 antecedents. Although FBDT is competitive in this respect, the rules constructed by this method are more difficult to interpret than those of CFM-BD, since the number of fuzzy sets used for each variable depends on the variable itself and is usually greater than 10. In CFM-BD, each variable is modeled with 5 fuzzy sets that are adjusted to the actual distribution of the variable, obtaining accurate and interpretable rules. In addition to interpretability, CFM-BD has been shown to be competitive in terms of classification performance, providing state-of-the-art discrimination capability.

REFERENCES

[1] H. Ishibuchi, T. Nakashima, and M. Nii, *Classification and modeling with linguistic information granules: Advanced approaches to linguistic Data Mining*. Springer-Verlag, 2004.

[2] K. Sarkar, P. Chatterjee, and N. R. Pal, "Finding Synergy Networks From Gene Expression Data: A Fuzzy-Rule-Based Approach," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 6, pp. 1488–1499, 2016.

[3] J. Sanz, M. Galar, A. Jurio, A. Brugos, M. Pagola, and H. Bustince, "Medical diagnosis of cardiovascular diseases using an interval-valued fuzzy rule-based classification system," *Applied Soft Computing Journal*, vol. 20, pp. 103–111, 2013.

[4] P. Singh, N. R. Pal, S. Verma, and O. P. Vyas, "Fuzzy Rule-Based Approach for Software Fault Prediction," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 5, pp. 826–837, 2017.

[5] C. Tsang, S. Kwong, and H. Wang, "Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection," *Pattern Recognition*, vol. 40, no. 9, pp. 2373–2391, 2007.

[6] M. Antonelli, D. Bernardo, H. Hagras, and F. Marcelloni, "Multiobjective Evolutionary Optimization of Type-2 Fuzzy Rule-Based Systems for Financial Data Classification," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 2, pp. 249–264, 2017.

[7] T. Nakashima, G. Schaefer, and Y. Yokota, "A weighted fuzzy classifier and its application to image processing tasks," *Fuzzy Sets and Systems*, vol. 158, no. 3, pp. 284–294, 2007.

[8] X. Zhang, E. Onieva, A. Perallos, E. Osaba, and V. C. Lee, "Hierarchical fuzzy rule-based system optimized with genetic algorithms for short term traffic congestion prediction," *Transportation Research Part C: Emerging Technologies*, vol. 43, no. Part 1, pp. 127–142, 2014.

[9] J. Gantz and D. Reinsel, "THE DIGITAL UNIVERSE IN 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East," 2012, http://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf. [Online]. Available: http://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf

[10] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[11] S. Owen, R. Anil, T. Dunning, and E. Friedman, *Mahout in Action*. Greenwich, CT, USA: Manning Publications Co., 2011.

[12] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar, "MLlib: Machine learning in Apache Spark," *Journal of Machine Learning Research*, vol. 17, 2016.

[13] P. Ducange, F. Marcelloni, and A. Segatori, "A MapReduce-based fuzzy associative classifier for big data," in *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2015, pp. 1–8.

[14] M. Elkano, M. Galar, J. Sanz, and H. Bustince, "CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems," *Fuzzy Sets and Systems*, 2017, In Press.

[15] A. Fernández, S. del Río, and F. Herrera, "A First Approach in Evolutionary Fuzzy Systems based on the lateral tuning of the linguistic labels for Big Data classification," in *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2016, pp. 1437–1444.

[16] A. Fernandez, E. Almansa, and F. Herrera, "Chi-Spark-RS: An Spark-built evolutionary fuzzy rule selection algorithm in imbalanced classifi-

cation for big data problems," in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2017, pp. 1–6.

[17] A. Ferranti, F. Marcelloni, A. Segatori, M. Antonelli, and P. Ducange, "A distributed approach to multi-objective evolutionary generation of fuzzy rule-based classifiers from big data," *Information Sciences*, vol. 415-416, pp. 319–340, 2017.

[18] A. M. García-Vico, P. González, M. J. del Jesus, and C. J. Carmona, "A first approach to handle fuzzy emerging patterns mining on big data problems: The EvAEFP-spark algorithm," in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2017, pp. 1–6.

[19] V. López, S. del Río, M. Benítez, and F. Herrera, "Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data," *Fuzzy Sets and Systems*, vol. 258, no. 0, pp. 5–38, 2015.

[20] F. Pulgar-Rubio, A. Rivera-Rivas, M. Pérez-Godoy, P. González, C. Carmona, and M. del Jesus, "MEFASD-BD: Multi-objective evolutionary fuzzy algorithm for subgroup discovery in big data environments - A MapReduce solution," *Knowledge-Based Systems*, vol. 117, no. Supplement C, pp. 70–78, 2017.

[21] A. Segatori, F. Marcelloni, and W. Pedrycz, "On Distributed Fuzzy Decision Trees for Big Data," *IEEE Transactions on Fuzzy Systems*, vol. PP, no. 99, pp. 1–1, 2017.

[22] A. Segatori, A. Bechini, P. Ducange, and F. Marcelloni, "A Distributed Fuzzy Associative Classifier for Big Data," *IEEE Transactions on Cybernetics*, vol. PP, no. 99, pp. 1–14, 2017.

[23] J. E. Angus, "The Probability Integral Transform and Related Results," *SIAM Review*, vol. 36, no. 4, pp. 652–654, 1994.

[24] C. P. Quesenberry, *Probability Integral Transformations*. John Wiley & Sons, Inc., 2004.

[25] N. U. Nair, P. G. Sankaran, and N. Balakrishnan, *Quantile-Based Reliability Analysis*. New York, NY: Springer New York, 2013, ch. Quantile Functions, pp. 1–28.

[26] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499.

[27] L. J. Eshelman, "The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination," ser. Foundations of Genetic Algorithms, G. J. Rawlins, Ed. Elsevier, 1991, vol. 1, no. Supplement C, pp. 265–283.

[28] D. DeWitt and J. Gray, "Parallel Database Systems: The Future of High Performance Database Systems," *Communications of the ACM*, vol. 35, no. 6, pp. 85–98, 1992.

[29] P. Jogalekar and M. Woodside, "Evaluating the Scalability of Distributed Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 6, pp. 589–603, 2000.

[30] H. Ishibuchi and T. Yamamoto, "Rule Weight Specification in Fuzzy Rule-Based Classification Systems," *IEEE Transactions on Fuzzy Systems*, vol. 13, no. 4, pp. 428–435, 2005.

[31] O. Cordón, M. del Jesus, and F. Herrera, "A proposal on reasoning methods in fuzzy rule-based classification systems," *International Journal of Approximate Reasoning*, vol. 20, no. 1, pp. 21–45, 1999.

[32] S. Ghemawat, H. Gobioff, and S. Leung, "The Google File System," in *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, ser. SOSP '03. ACM, 2003, pp. 29–43.

[33] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'12, 2012, pp. 2–2.

[34] J. C. Gámez, D. García, A. González, and R. Pérez, "On the use of an incremental approach to learn fuzzy classification rules for big data problems," in *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2016, pp. 1413–1420.

[35] R. Romero-Zaliz, A. González, and R. Pérez, "Incremental fuzzy learning algorithms in big data problems: A study on the size of learning subsets," in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2017, pp. 1–6.

[36] A. Fernández, C. J. Carmona, M. J. del Jesus, and F. Herrera, "A View on Fuzzy Systems for Big Data: Progress and Opportunities," *International Journal of Computational Intelligence Systems*, vol. 9, pp. 69–80, 2016.

[37] D. Whitley, S. Rana, J. Dzubera, and K. E. Mathias, "Evaluating evolutionary algorithms," *Artificial Intelligence*, vol. 85, no. 1, pp. 245–276, 1996.

[38] J. Alcalá-Fdez, R. Alcalá, and F. Herrera, "A Fuzzy Association Rule-Based Classification Model for High-Dimensional Problems With Genetic Rule Selection and Lateral Tuning," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 5, pp. 857–872, 2011.

[39] J. Sanz, A. Fernández, H. Bustince, and F. Herrera, "IVTURS: A Linguistic Fuzzy Rule-Based Classification System Based On a New Interval-Valued Fuzzy Reasoning Method With Tuning and Rule Selection," *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 3, pp. 399–411, 2013.

[40] R. Barandela, J. Sánchez, V. García, and E. Rangel, "Strategies for learning in class imbalance problems," *Pattern Recognition*, vol. 36, no. 3, pp. 849–851, 2003.

[41] L. Eshelman and J. Schaffer, "Real-coded genetic algorithms and interval schemata," *Foundations of Genetic Algorithms*, vol. 2, pp. 187–202, 1993.